PROGRAMACION EN BASIC (I)

NUEVAS TECNOLOGIAS

BIBLIOTECA DE ELECTRONICA/INFORMATICA



marcombo



BIBLIOTECA DE ELECTRONICA/INFORMATICA

PROGRAMACION EN BASIC (I)



Esta obra es una nueva edición actualizada y ampliada de la obra originalmente publicada por Marcombo, S.A. de Boixareu editores, con el título de «Aplicaciones de la Electrónica»

El contenido de la presente obra ha sido realizado por Marcombo, S.A. de Boixareu editores, bajo la dirección técnica de José Mompin Poblet, director de la revista «Mundo Electrónico»

ISBN 84-7634-485-6 (Obra completa)
ISBN 84-7634-809-6 (Vol. 47)
D. L.: B. 35621-1986

Impreso y encuadernado por
printer industria gráfica, sa c.n. II, cuatro caminos, s/n
08620 sant vicenç dels horts barcelona 1986

Printed in Spain

Programación en Basic (I)

INTRODUCCION

A medida que avanzan la ciencia y la técnica, va miniaturizándose el soporte material de la información. Desde los primeros jeroglíficos egipcios, pasando por las toscas letras de la imprenta medieval hasta el videodisco de nuestros días, se han superado ingentes etapas para reducir



Computador personal PX-8 de Epson. El BASIC es uno de los lenguajes más utilizados para dialogar con este tipo de computadores. Su tamaño permite trabajar con ellos en cualquier lugar, ya que admiten baterías auxiliares.

el tamaño del soporte de toda clase de información. Esta reducción ha pasado desde codificar el lenguaje por medio de figuras, luego con letras, hasta llegar finalmente al sistema binario, en que la información se codifica en bits equivalentes a un cero y a un uno o a presencia y ausencia de señal eléctrica.

Con las figuras de los jeroglíficos se formaba un lenguaje y

lo mismo podríamos decir de las letras y los bits, lenguaje que permite toda clase de información o representación, desde un hecho histórico de hace siglos hasta un concepto matemático como las ternas pitagóricas que ya utilizaban los egipcios. Y como todo lenguaje tiene una serie de cualidades y reglas para representar la información, en este libro se va a tratar de las cualidades y reglas del lenguaje más utilizado en la Informática, el lenguaje BASIC.

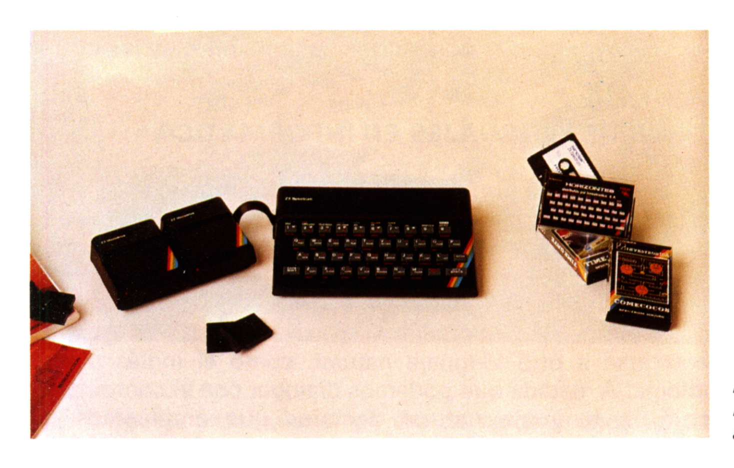


Computador personal de bolsillo, a pilas o con la tensión de la red. El reducido tamaño queda compensado por la función múltiple que puede desarrollar cada tecla.

La Informática se caracteriza por la facilidad en el tratamiento o en el proceso de la información. El avance en este campo podemos comprenderlo si comparamos las dificultades que presentaba una multiplicación cuando se utilizaban los números *romanos* y esta misma operación hecha con cualquier medio de los que nos brinda la Informática actual, como son una calculadora de bolsillo o un computador personal. Pero no hay que olvidar que en ambos casos se utiliza un lenguaje con reglas precisas. No se pueden poner cuatro letras iguales seguidas en la numeración *romana* para representar, por ejemplo, la cifra 4, y lo

mismo podríamos decir de cualquiera de los más de 1.000 lenguajes que se han desarrollado para la Informática. Utilizando un paralelismo diremos que estas reglas precisas constituyen la *sintaxis del lenguaje*, sintaxis que tiene que verificar el computador para procesar correctamente los datos que le introducimos.

Tanto la comprobación de la sintaxis del lenguaje usado como el mismo proceso de los datos que se expresan con este lenguaje, no podrían existir sin el estado actual de la electrónica. Si se busca miniaturizar el soporte de la información y su proceso es para poder trabajar más rápidamente. Piénsese lo que significa el obtener cifras del número «pi» (π) desde que se lo propusieron hace siglos y llegaron a 707 decimales, haciendo operaciones durante 20 años, pasando por la obtención de las mismas cifras con una calculadora de bolsillo en BASIC como la FX-702P de CASIO que, a plena capacidad, llega a obtener 885 decimales



El computador ZX Espectrum junto con sus accesorios más comunes.

trabajando continuamente durante 98 horas, hasta la rapidez que se obtiene con los modernos computadores construidos con elevada densidad de integración en sus chips que realizan la operación en segundos. Pero lo más curioso de este ejemplo es que el heroico matemático de hace siglos que empleó 20 años se equivocó a partir del decimal número 528, error que no fue detectado hasta 1945, cuando ya se podía disponer de equipos electrónicos adecuados.

Los lenguajes de programación hacen posible el estado actual del proceso de datos. Cada lenguaje se crea para una finalidad determinada, y si el BASIC es utilizado en la mayor parte de los computadores personales es por su simplicidad y porque se le han añadido nuevas órdenes a las del lenguaje original, de tal modo que sirve para una gran cantidad de aplicaciones.

Este lenguaje fue creado en 1964 por John Kemey y Thomas Kurtz en el Darmouth College de New Hampshire, con la finalidad de introducir a los jóvenes estudiantes de todo tipo en la Informática. Sus iniciales se tomaron de las palabras *B*eginners *A*ll-purpose *S*ymbolic *I*nstruction *C*ode, que viene a significar: *código de las instrucciones simbólicas de tipo general para principiantes.* Pero la facilidad de utilización y aprendizaje junto con su simplicidad hicieron que la mayoría de constructores de computadores lo adoptasen, de tal forma que es el lenguaje más usado en la actualidad y se piensa que pronto formará parte de las calculadoras de bolsillo programables, sustituyendo los anteriores lenguajes que utilizan las funciones de las teclas.

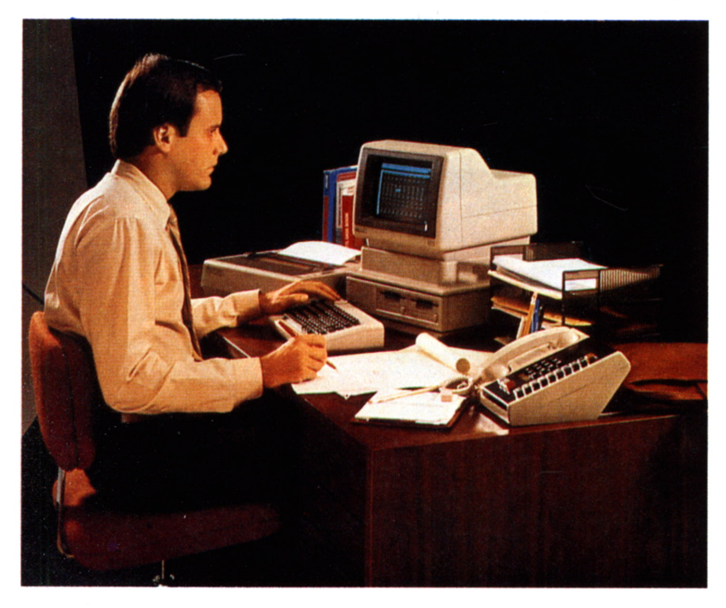
TIPOS DE LENGUAJES EN INFORMATICA

Todos los computadores procesan la información, que les entra en código binario, en bits. Este lenguaje, denominado también *lenguaje de máquina*, se representa con ceros y unos, por lo que es difícil el proceso de funcionamiento y se presta a muchos errores al tener solamente dos dígitos.

Se busca en la actualidad apartarse de este lenguaje y acercarse a otro lenguaje natural, como el inglés u otro idioma. A medida que podemos dialogar con el computador con un lenguaje natural decimos que empleamos un *lenguaje de alto nivel*. Este tiene que ser sencillo de aprender y debe ser innecesario conocer cómo lo utiliza el computador en sus procesos, al revés de lo que ocurre con el lenguaje de máquina, que se denomina de *bajo nivel* porque tiene que utilizarse sabiendo cómo lo empleará el computador.

Debido a que el lenguaje de máquina tiene sus ventajas, se utiliza un lenguaje algo mejorado que se llama *lenguaje ensamblador*. Una de las ventajas mencionadas puede ser que funciona mucho más rápido que el lenguaje de alto nivel, porque no hay que olvidar que todo lenguaje tiene que

ser pasado al de máquina para que lo entienda el computador. Otra ventaja que tiene es que permite con facilidad dejar privado un programa y con ello no es posible reproducirlo. Este extremo es importante porque asegura que el trabajo de un programador, muchas veces considerable, puede pagarse a un precio adecuado si no se pueden copiar los programas, generalmente suministrados con discos o con cintas cassettes.

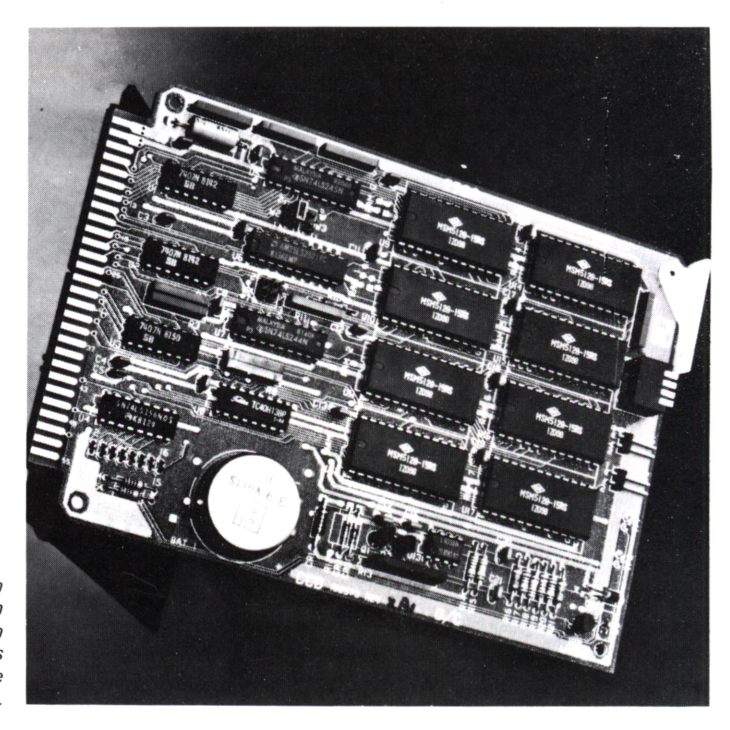


Microcomputador con capacidad para procesado de textos y gestión de una base de datos, permitiendo además la accesibilidad a otros computadores remotos.

El lenguaje ensamblador en vez de ceros y unos utiliza *símbolos* de pocas letras y con ello permite una mejor lectura, aunque por ser de bajo nivel está determinado por el microprocesador que tiene el computador y por el funcionamiento de su proceso interno.

Los lenguajes de alto nivel permiten concentrarse en las funciones del programa, dejando para el computador la tarea de cómo procesar este programa. Según sea la forma de pasar a lenguaje de máquina, los lenguajes de alto nivel pueden ser interpretados o compilados. En los lenguajes interpretados como el BASIC cada una de sus órdenes pasa

a código de máquina al funcionar, realizándose este paso por una parte del computador que se llama *intérprete del BASIC*. Pero en los *lenguajes compilados*, como el FORTRAN, ALGOL, COBOL, PASCAL, etc. se pasa una sola vez el programa al lenguaje de máquina compilándolo y por ello



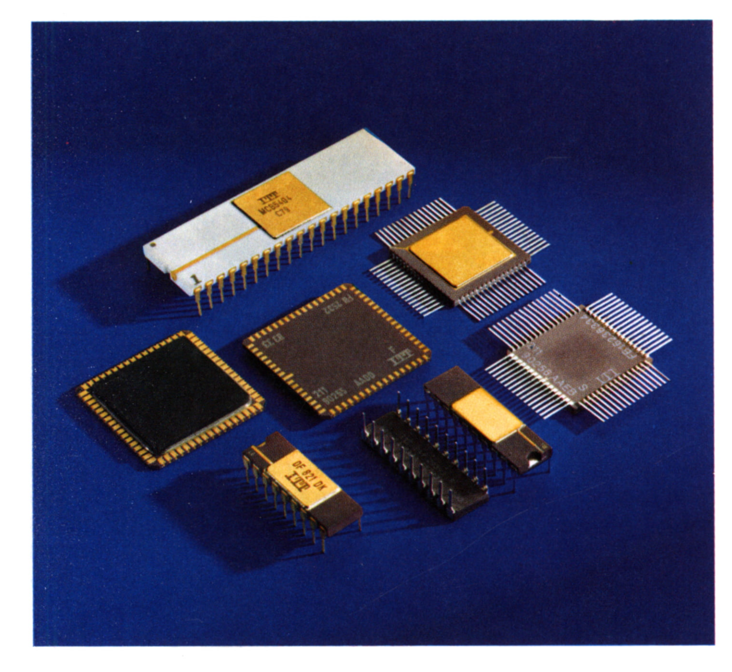
Tarjeta de un microcomputador en donde se observan diversos circuitos integrados y memorias de semiconductor.

pueden ser más rápidos. Esta rapidez que les confiere ventaja, presenta también el inconveniente de que no se pueden modificar sin tenerlo que volver a compilar, mientras que el BASIC y los demás lenguajes interpretados pueden modificarse sobre la marcha y con ello puede comprobarse el resultado que da el funcionamiento del programa cuando se introducen modificaciones. Podríamos decir que permite una rápida comprensión del programa por la facilidad de introducir modificaciones.

La mayor parte de la gente que utiliza los computadores personales no es experta en Informática y necesita aprender a programar viendo el resultado que se obtiene al introducir modificaciones. Y aunque solamente utilicen programas que se encuentran en libros y revistas especializadas, si el programa no funciona se puede buscar la causa del error al hacer pequeñas modificaciones, lo cual es bastante lento en lenguajes compilados.

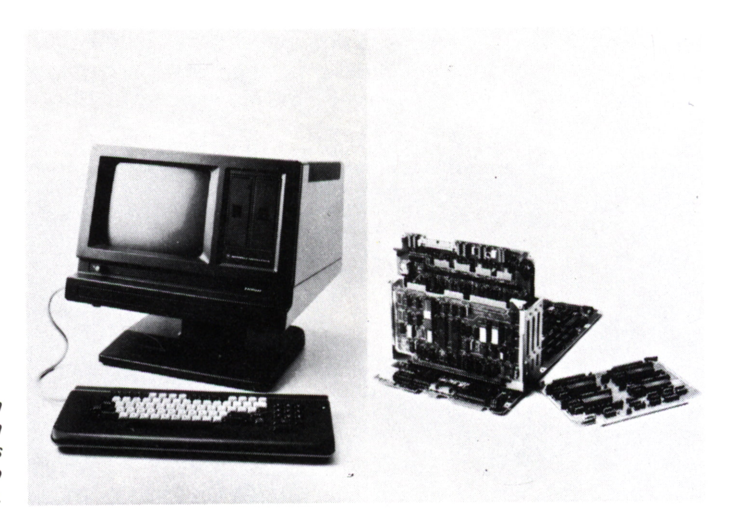
CUALIDADES DEL LENGUAJE BASIC

La facilidad de aprender este lenguaje, junto a que tiene pocas órdenes y funciones, hace que sea lo que se propusieron sus creadores: un lenguaje para los que no tienen conocimientos de Informática pero desean programar sin complicaciones, y en el caso de que éstas aparezcan, un lenguaje que pueda modificarse o ensayar otras órdenes y funciones, sin dificultades y sin tener que esperar mucho.



Diferentes tipos de encapsulados de los circuitos integrados preparados para formar las unidades de memoria y otras secciones de los computadores personales

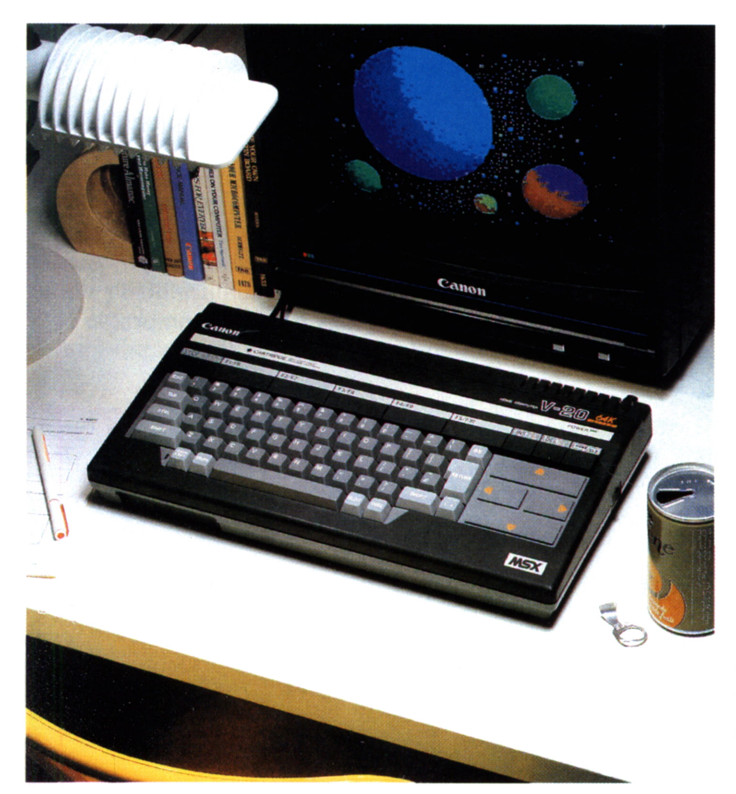
A medida que se han creado nuevos computadores personales, el lenguaje BASIC se enriquece con nuevas órdenes con lo que se persigue un mejor funcionamiento, pero por suerte hay una gran mayoría de órdenes primitivas del lenguaje que permiten programar fácilmente cada nuevo computador que aparece en el mercado. Con ello podríamos considerar que aparecen dialectos del lenguaje original, sobre todo en lo que hace referencia a la forma de presentar los datos en la pantalla al trabajar en alta resolución o en modo gráfico, y también en la forma de entrarle los datos con un disco magnético. Entonces hay que recurrir al manual de funcionamiento del computador para entrar en los detalles de su manejo, aunque ya se sabe que las generalidades del BASIC serán las de los demás computadores.



Computador personal con pantalla, destacando, a la derecha, las tarjetas modulares que lo forman.

Se podrían resumir en las siguientes las principales variantes a las que hay que prestar atención para hacer funcionar un computador en lenguaje BASIC:

1) Hay que comprobar si en cada línea del programa hay un único grupo de *órdenes* y *funciones* que forman lo que se llama una *sentencia* o *instrucción*, ya que muchos computadores pueden tener varias sentencias por línea, separadas entonces por dos puntos o por otro signo como el &.



Los computadores personales están preparados para trabajar en un lenguaje sencillo, como es el BASIC. (Cortesía: Canon).

2) Hay que conocer el número de caracteres alfabéticos o numéricos que puede contener una *línea de programa,* sobre todo si puede tener varias sentencias.

3) Es importante comprobar la puntuación a usar en lo referente a las posibilidades aditivas del punto y coma, de la coma y de las comillas.

4) Hay que tener a mano la equivalencia de los *códigos de error* que aparecerán inmediatamente cuando se empiece a programar, pues hay computadores que los especifican con palabras escritas generalmente en inglés, pero otros tienen gran riqueza de ellos en forma de una clave que necesita su interpretación. Un buen computador indica no solamente la línea en que ha encontrado un error de sintaxis, sino que lo localiza con algún signo que parpadea, lo cual

permite aprender rápidamente. No es lo mismo que aparezcan en la pantalla un par de números de los que uno indica el número de línea y el otro el tipo de error, a que aparezca el mensaje de «valor ilegal» y la señal del cursor parpadeando sobre el signo de la raíz cuadrada, por lo que comprenderemos en seguida que el programa ha presentado una variable negativa para extraerle la raíz cuadrada.

5) La precisión con que trabaja el computador está relacionada con el número de cifras máximo que procesa. A veces es de 8, en otros llega a 10 y utilizando la doble precisión se alcanzan las 15 cifras; hay que comprobar si la última de éstas queda redondeada o no. Posteriormente se tratará del tema de los errores en las operaciones aritméticas que pueden hacer fracasar la programación de un caso sencillo.



Para poder manejar el volumen de datos que intervienen en un pequeño negocio, puede ser suficiente utilizar un computador de reducidas prestaciones. (Cortesía: Philips).

6) Si se va a utilizar el BASIC de un computador para representaciones gráficas, hay que comprobar el número máximo de puntos de la pantalla que pueden utilizarse, así como toda la serie de facilidades que dan los comandos específicos para las representaciones gráficas, sea en color o no. Y lo mismo podría decirse de otras posibilidades del

computador personal, como la síntesis de música o la síntesis de voz, que ya empiezan a divulgarse en el mercado.

7) Finalmente, y también dentro de las cualidades que le pedimos al lenguaje BASIC del computador personal, hay que comprobar si es posible por programa mandar los equipos de máquinas que funcionan con relés. Si es posible, cualidad que cada vez es más frecuente en el mercado, hay que asegurarse si se realiza con órdenes de BASIC o tiene que utilizarse su lenguaje ensamblador, lo cual ya no es tan sencillo como usar un lenguaje de alto nivel.

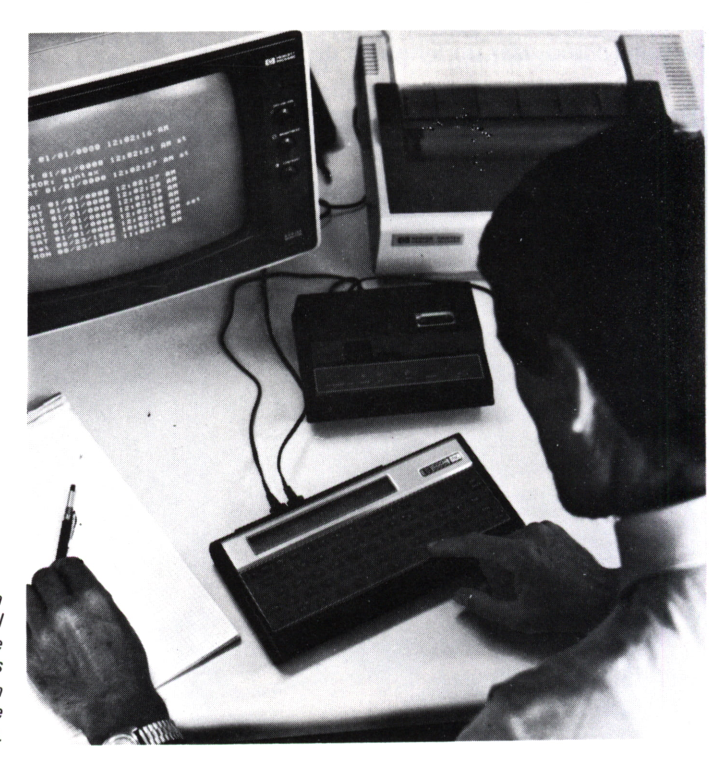


Los computadores personales pueden trabajar también con información almacenada en microdiscos flexibles.

La sintaxis del lenguaje BASIC

En primer lugar hay que distinguir entre órdenes y funciones del lenguaje BASIC, términos que a veces se confunden porque ambos tienen por misión hacer que el programa funcione. La diferencia está en las órdenes para el interpretador del BASIC a fin de que ejecute una o varias instrucciones o sentencias, dentro de las cuales puede haber

funciones matemáticas diversas. En general las instrucciones o sentencias van precedidas por el número de la línea del programa, pero esto no ocurre para las órdenes del tipo RUN, LIST, SAVE, BREAK, etc. No obstante hay manuales de computadores que a todos los grupos de letras reservados para el funcionamiento del programa les llaman órdenes, sean funciones o no.

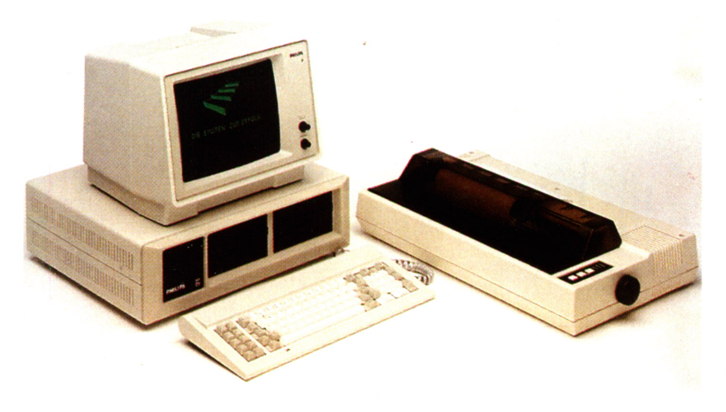


Hewlett Packard presenta un computador personal adaptado al lenguaje BASIC con posibilidades de interconexión con un buen número de periféricos.

Estos grupos de letras, reservados para el funcionamiento del programa, se tienen que usar adecuadamente, de lo contrario aparecerá el mensaje de error correspondiente. Así, cuando el intéprete del BASIC encuentra la palabra SIN, intenta calcular el seno del correspondiente ángulo y si se utiliza este grupo de letras como una variable se detiene el programa.

En la programación del lenguaje BASIC se utiliza primero un número de línea y luego una serie de órdenes, funciones y signos que forman una o varias instrucciones o sentencias. Las cifras que se pueden utilizar van del 0 al 9, a veces distinguiendo el cero de la letra «o» poniéndole una raya transversal. Como letras se utilizan las del alfabeto inglés, de la A a la Z, pero sin la Ñ ni la CH.

Los símbolos aritméticos utilizados son + − * / ↑. Obsérvese que la multiplicación utiliza el asterisco y la potenciación una flecha o el ángulo de su parte superior, aunque a veces las impresoras utilizan para la potenciación el doble asterisco.



Computador Philips que incluye el teclado, lectores de floppy disc, pantalla e impresora.

Como caracteres de separación se utilizan la coma, el punto y coma y los dos puntos. En las comparaciones entre grupos de datos, sean constantes, sean variables o grupos de ellas denominadas expresiones, los signos son: = < > < = > = < >. El último signo se utiliza para indicar que no es igual.

Los comandos del tipo matemático se denominan *funciones* y los más corrientes son: las trigonométricas SIN, COS, TAN, ATAN, esta última es el arco tangente en radianes. La raíz cuadrada tiene por función SQR y para los logaritmos decimales se utiliza la funcion LOG, mientras que para los

neperianos se emplea LN. La función exponencial se utiliza como EXP.

En los tratamientos numéricos se utilizan las funciones INT para la parte entera de un número, la función ABS para tener el valor absoluto y la función SGN, que si el valor es nulo da como resultado cero, mientras que si es menor o mayor que cero da los valores más uno y menos uno.

Quedan otras funciones importantes, como la RND que proporciona números al azar si es que el computador tiene generador aleatorio propio, de lo contrario se verá la forma de obtener estos números pseudo-aleatorios por una sencilla línea de programa.



Para el perfecto dominio de cualquier computador resulta imprescindible conocer el lenguaje de programación y las instrucciones de manejo.

Algunos computadores tienen la constante «pi» con un número de cifras ya programado, pero no suele haber otras constantes. Como variables se utilizan las letras del alfabeto, ya sean solas o con subíndice encerrado entre paréntesis, como por ejemplo A(8), que puede contener en vez del número constante 8 otra variable, por ejemplo A(J), o también grupos de variables constituyendo una expresión, ejemplo A(2 – B).

Si se trata de variables que contienen cadenas de caracteres de texto, denominadas *strings*, se utilizan las letras del alfabeto junto con el signo dólar. Ejemplo A$(3)=«ABCDF& %()zx». Entonces hay que encerrar la cadena de caracteres entre comillas, por lo que las comillas no pueden entrar como caracteres pero los otros signos sí.



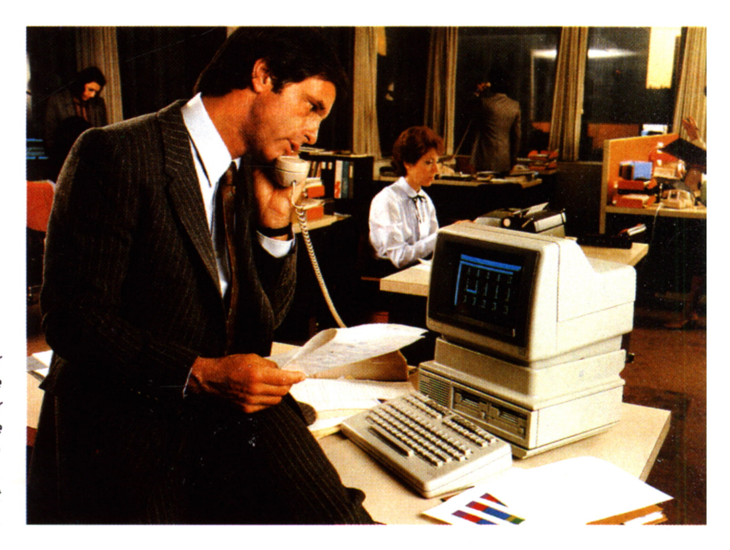
Para un perfecto aprovechamiento de las posibilidades de trabajo que brinda un computador es conveniente disponer del mayor número posible de programas. (Cortesía: DSE-Apricot).

Hay un conjunto de órdenes que hacen funcionar el programa, las cuales se detallan a continuación; su número puede aumentar a medida que un computador tiene un BASIC más potente.

Una vez escrito el programa se pondrá en marcha con RUN. Esta orden generalmente deja a cero las variables. Pero antes de escribirlo es conveniente borrar el programa anterior, lo cual se hace con NEW, de lo contrario podrían quedar líneas del programa anterior que seguirían funcionando con el nuevo programa.

Si se desea visualizar el programa que queda en la pantalla con LIST y si se desea imprimirlo puede usarse la orden LLIST. Para guardarlo en una cinta tipo cassette se emplea SAVE y para volverlo a cargar en el computador se usa LOAD. Estas órdenes hay computadores que las denominan CSAVE y CLOAD.

Si cuando el programa está funcionando deseamos detenerlo, la orden usual es BREAK, pero si se hace desde el propio programa se utiliza STOP.



Terminal de computador potente, HP 125, de sobremesa, capaz de ser conectado a una base de datos a través de un HP-300. (Cortesía: Hewlett Packard).

Finalmente hay que indicar que al final del recorrido del programa conviene poner un END, que no es necesario esté al final del listado. Pero la sintaxis, además de letras y signos, tiene unas reglas que se expondrán en los apartados siguientes.

LA PROGRAMACION

Cuando hay que repetir un proceso muchas veces vale la pena que esta repetición sea automática o *programada*. Un programa puede venir mandado por una cinta perforada, por una secuencia de contactos que proporciona un tambor giratorio o bien por otro tipo de secuencia como la que tienen los computadores, que es una secuencia de instrucciones proporcionada por circuitos electrónicos. Las cintas perforadas durante muchos años han programado las máquinas textiles. Los tambores que accionan contactos y válvulas neumáticas se han usado en muchas máquinas

automáticas, pero la electrónica ha proporcionado grandes ventajas a la programación.

Con la rapidez de funcionamiento de la electrónica se puede bifurcar fácilmente a diferentes pasos del programa, lo cual no es posible si el programa está grabado en un tambor giratorio o en una cinta perforada, al menos hay muchas dificultades. Otra de las ventajas de la electrónica está en que podemos modificar o alterar los pasos de la programación, añadir nuevas condiciones y si es necesario tener un número elevado de estas condiciones de la programación.



Aprovechando las posibilidades que brinda un programa, resulta sencillo verificar el comportamiento de un proceso industrial, modificando las variables de cada programa. (Cortesía: Husky Computers).

La idea de la programación es antigua. Se trata de que un dispositivo vaya haciendo una serie de operaciones, una después de otra y siempre las mismas. En este sentido, uno de los primeros programas fue el reloj mecánico que da vueltas haciendo siempre lo mismo, toca las campanadas en el momento preciso y en un número exacto. Pero por ser mecánico tiene el inconveniente de no poderse modificar y por ello es un programa que sólo sirve para dar las horas.



Conjunto completo de un computador y periféricos de la firma Loewe. (Cortesía: Visualdata).

La electrónica ha permitido la gran versatilidad de la programación actual. Podemos dar la hora con un computador personal que tenga salida musical y a la vez podemos calcular cualquier cosa con facilidad, si se quiere podemos ponerle órdenes de comparación de manera que nos diga cuál es la próxima jugada que hay que hacer en una partida de ajedrez con un nivel de exploración dado de las próximas jugadas, y con salidas adecuadas puede desde el programa llevarse el control de un equipo de relés que controla los motores de una máquina eléctrica.

El programa puede considerarse como una serie de casillas que el computador recorrerá haciendo lo que tenga escrito en cada una de ellas. Si en la primera casilla hay la indicación de que el radio vale 2, le pondremos LET R=2; también puede omitirse el LET quedando solamente R=2. Si en la segunda casilla ponemos la fórmula del área del círculo, escribiremos A=PI*R*R. Si en la tercera queremos que el computador nos dé el valor del cálculo pondríamos PRINT A

y con ello quedaría en la pantalla el resultado de la programación: 12.56637062. Como hay que dar por terminado el proceso, en la última casilla pondríamos END. Un programa de este tipo tendría el aspecto del programa n.º 1 de la figura 18.

```
10 R=2
20 A=PI*R*R
30 PRINT A
40 END
```

Figura 18. Listado del programa n.º 1.

Cada casilla está representada por una línea que tiene su numeración creciente. En este caso hay en cada línea una única sentencia o instrucción, pero hay computadores que permiten poner varias, por lo que el programa podría tener solamente un par de líneas, como el programa n.º 2 de la figura 19 en el que se separan las instrucciones con los dos puntos.

Las líneas del programa se suelen numerar de diez en diez para poder intercalar otras líneas al modificarlo, pero pueden numerarse de cinco en cinco o de otra forma mientras sea con numeración creciente, aunque tenga varias cifras cada número de línea.

```
10 R=2:A=PI*R*R:PRINT A
20 END
```

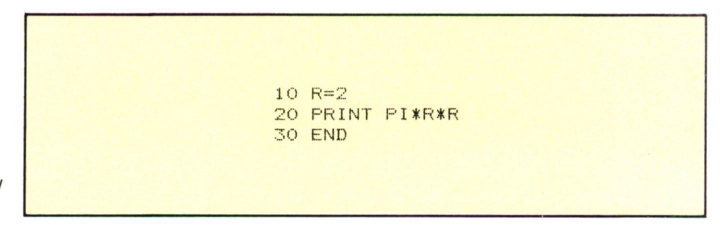
Figura 19. Listado del programa n.º 2.

En el anterior programa hay dos constantes: la cifra 2 y el valor de «pi», mientras que son variables las letras R y A. Se ha usado para la multiplicación el signo del asterisco, pero respecto al signo igual hay que tener en cuenta que en

BASIC indica que la parte derecha se asigna a la izquierda del signo.

Por esto en la izquierda hay una sola variable ya que el computador no sabría qué hacer si encontrase una instrucción como A+B=3−R, puesto que asignaría el valor de 3−R al valor A+B y, como hay infinitas soluciones, daría un mensaje de error.

Para ilustrar cómo se asigna en BASIC, puede verse la expresión A=A+1, que se emplea frecuentemente para contar las veces que se pasa por esta línea del programa. Al ejecutarla el interpretador del lenguaje BASIC, al valor de la variable A de la derecha de la igualdad se le sumará 1 y el resultado se asignará al nuevo valor de la variable A. Con ello tendremos las veces que se hará esta asignación y es frecuente que esta sencilla instrucción sea utilizada como contador.



```
10 R=2
20 PRINT PI*R*R
30 END
```

Figura 20. Listado del programa n.º 3.

La salida de datos por la línea 30 del primer programa indica al computador que ponga el resultado del cálculo del área en la pantalla, valor que está en la variable A, pero como la orden PRINT puede llevar a continuación una constante, una variable o un conjunto de variables formando una expresión, se podría reducir el programa al representado en la figura 20, programa n.º 3, en el cual no se utiliza la variable A.

Algunos computadores necesitan que todas las variables tengan previamente a su utilización la declaración de su valor, aunque éste sea cero. En otros al empezar el programa a funcionar por RUN, quedan todas las variables a cero y no hay que declararlas. Hay que consultar el libro de instrucciones del computador para poder programar bien.

Si continuamos viendo las reglas que presenta el funcio-

namiento del lenguaje, o mejor dicho su sintaxis, veremos que en el programa anterior no hay problemas sobre el tipo de operación que ha de hacer el intérprete en primer lugar, ya que se trata solamente de multiplicaciones, pero si hay distintas operaciones en la expresión hay que atenerse a una prioridad jerárquica de las operaciones. Por ejemplo, si utilizamos tres variables con valores A=3, B=2, C=5, al utilizarlas en la siguiente sentencia D=A/B/C queda poco claro cuál será el resultado, ya que puede ser D=(A/B)/C=(3/2)/5=0,3 o bien D=A/(B/C)=3/(2/5)=7·5. El uso de los paréntesis es un gran auxiliar en la programación en BASIC, de lo contrario tiene que fijarse la prioridad de las operaciones, lo cual no sirve en este caso por haber dos operaciones iguales. Este dilema suelen resolverlo los computadores operando de izquierda a derecha, sistema que daría el primer resultado.

Y lo mismo podría decirse si en la expresión hay operaciones que tienen la misma prioridad jerárquica, por ejemplo si se asignan a las variables B=3, C=−3 la siguiente expresión tendrá por valor: A=B*B/C*C=9. Pero valdrá 1 si lo que se quería expresar es A=B*B/(C*C).



Al manejar los computadores debe tenerse siempre la precaución de emplear el código adecuado ya que, de no hacerlo, la máquina no aceptará los datos propuestos. (Cortesía: Philips).

Las distintas prioridades de las operaciones hacen más fácil la programación, pero en caso de duda siempre hay el recurso de poner paréntesis con la condición de que tiene que haber tantos abiertos como cerrados y de que cada computador tiene un máximo de paréntesis pendientes de cerrar, lo cual hay que consultar con el manual de funcionamiento.

```
10 INPUT R
20 PRINT PI*R*R
30 END
```

Figura 22. Listado del programa n.º 4.

Ahora podemos pasar a modificar la entrada de datos del programa anterior, puesto que sólo serviría para el caso de que el radio valiese 2. Para ello utilizamos la orden de entrada de datos INPUT, tal como puede verse en el programa n.º 4 de la figura 22, al ponerlo en marcha por RUN el computador se detendrá presentando en la pantalla el signo de interrogación y esperando que le entren el valor

```
10 INPUT "RADIO=";R
20 PRINT PI*R*R
30 END
```

Figura 23. Listado del programa n.º 5.

del radio. Si hay que entrarle diversos valores es preferible que aparezca en la pantalla un mensaje adecuado para evitar errores. Esto podemos verlo en el programa n.º 5 de la figura 23, en el que se ha puesto entre comillas el mensaje de lo que hay que entrar, que es el caso más corriente, aunque hay computadores en los que este mensaje hay que ponerlo previamente con la orden PRINT.

Este programa lo iniciaríamos con RUN cada vez que deseásemos saber el área de un círculo, pero si lo tenemos que utilizar muchas veces seguidas es preferible que el propio programa vuelva al origen utilizando el comando GOTO... que equivale a «ir a la línea núm....». Entonces queda con una línea más y podemos verlo en el programa n.º 6 de la figura 24, mucho más práctico de utilizar, pero poco a poco se va complicando la programación en BASIC por lo que es necesario utilizar dos recursos que permiten al leer un programa, enterarse de cómo funciona y hallar las causas de cualquier detención inadecuada.

```
10 INPUT "RADIO=";R
20 PRINT PI*R*R
30 GOTO 10
40 END
```

Figura 24. Listado del programa n.º 6.

Un primer recurso es utilizar el comando REM que se emplea para poner en el programa comentarios distintos, los cuales son ignorados por el programa en el proceso de cálculo.

Esto lo podemos ver, aunque poniendo los comentarios al máximo, en el programa n.º 7 de la figura 25 para el mismo caso que se ha tratado anteriormente. En programas largos y para no ocupar memoria RAM del computador, (memoria

```
10 REM Programa que obtiene el area del circulo
20 REM ENTRADA DEL VALOR DEL RADIO
30 INPUT "RADIO=";R
40 REM CALCULO DEL RESULTADO
50 A=PI*R*R
60 REM VALOR EN PANTALLA
70 PRINT A
80 REM RETORNO AL ORIGEN DEL PROGRAMA
90 GOTO 20
100 END
```

Figura 25. Listado del programa n.º 7, equivalente al n.º 6.

variable utilizable por el usuario), se ponen los comentarios necesarios en cada grupo de operaciones para una fácil comprensión del programa.



Modelo de computador profesional de la firma Digital. Este computador está preparado para adaptarse plenamente al típico trabajo de las oficinas.

Otro recurso para facilitar la comprensión del programa es utilizar el diagrama de flujo del proceso que sigue el programa. En realidad hay que preparar previamente el programa con un *algoritmo* seguro y luego diseñar el correspondiente diagrama de flujo, antes de pasar a la programación línea por línea. Se denomina *algoritmo* al conjunto de reglas exactas y precisas que dan el orden en que se tiene que ejecutar un sistema de operaciones para resolver un problema dado en un tiempo finito y tener con ello un resultado único. En el caso que nos ocupa el algoritmo es muy sencillo, se trata de multiplicar el número «pi» por el cuadrado del radio y con ello tenemos el área del círculo. Pero si tuviésemos que hallar el radio del círculo dada su área, tendríamos que tener en cuenta que algunos valores no dan resultado, como por ejemplo los valores negativos. Ello nos obliga a poner en el programa un *filtro* que los elimine, lo cual veremos posteriormente.

Una vez tenemos seguro el algoritmo, el camino que seguirá el programa lo representamos con el *diagrama de flujo*. Este diagrama se representaba hace años con símbolos

que recordaban el material que se usaba entonces, fichas perforadas, papel de impresora, etc. pero como en la actualidad han cambiado algunos materiales, sobre todo la forma de entrar datos y programa con fichas perforadas, el diagrama de flujo se simplifica utilizando solamente 3 ó 4 símbolos: un círculo que indica el principio y fin del programa, un rombo para cuando hay bifurcaciones debidas a una condición y un rectángulo para las demás operaciones.

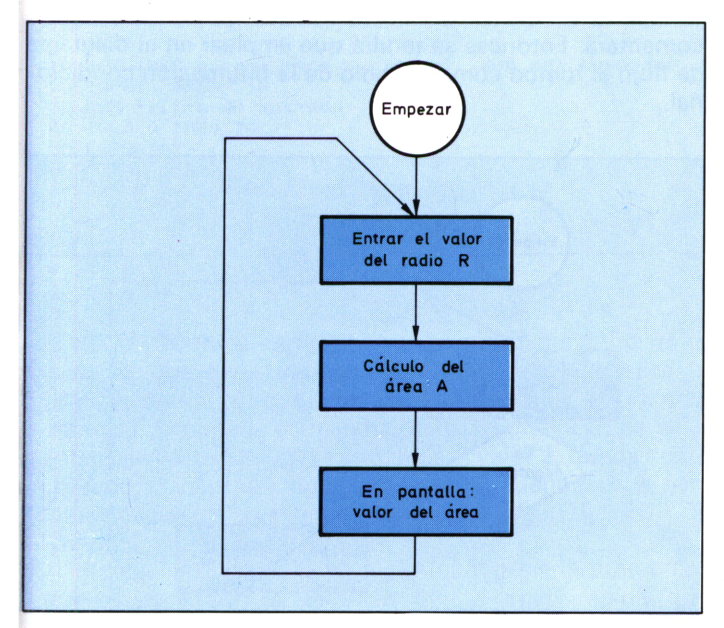


Figura 27. Diagrama de flujo correspondiente al programa n.º 7.

En la figura 27 podemos ver la equivalencia del programa n.º 7 con su diagrama de flujo, de manera que muchas veces el paso del citado diagrama al listado del programa se puede seguir con facilidad.

Un buen algoritmo tiene que tener en cuenta los errores que pueden producirse y con ello se evita una posible detención de la marcha del programa. Hay errores que ya se comprende que el computador no puede procesar, como por ejemplo dividir por cero, hallar raíces cuadradas de números negativos, en trigonometría la tangente de 90º, etc. Si al

entrar cualquier número por la orden INPUT no lo tenemos en cuenta, tendremos la correspondiente detención de la marcha del programa con el adecuado mensaje en la pantalla. Es necesario, por lo tanto, adelantarse poniendo un filtro en el programa.

Así, si queremos hacer un programa que nos calcule el lado de un cuadrado cuando nos dan su área, tendremos que extraer la raíz cuadrada de este área, pero para que no se le puedan entrar valores negativos le pondremos un filtro utilizando la orden IF... THEN... que posteriormente se comentará. Entonces se tendrá que emplear en el diagrama de flujo el rombo como símbolo de la bifurcación condicional.

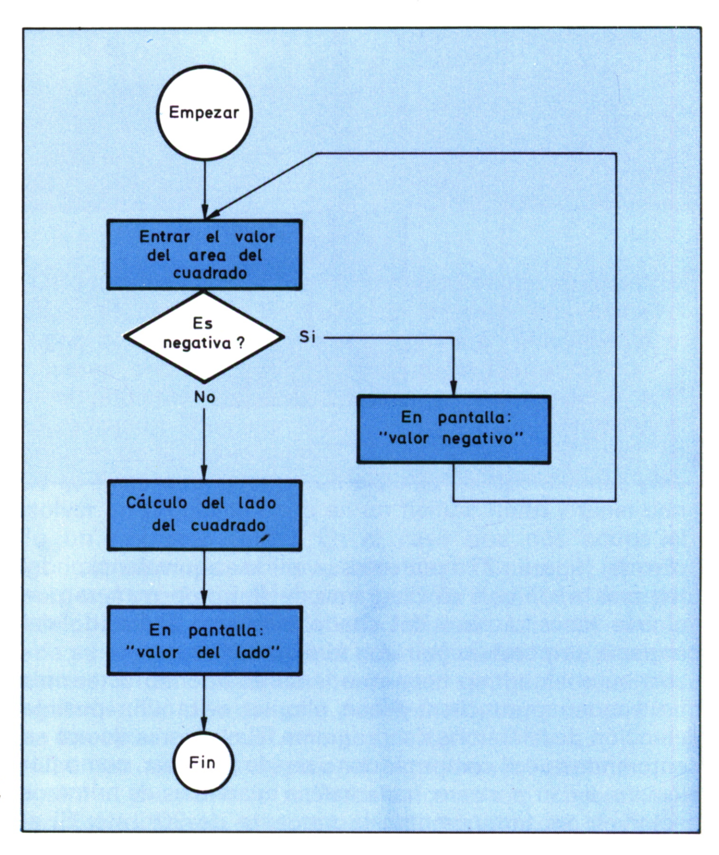


Figura 28. Diagrama de flujo correspondiente al programa n.º 8.

En la figura 28 podemos ver el diagrama de flujo del programa n.º 8, que calcula el lado de un cuadrado con la bifurcación condicional que se establece si se le entra un valor del área negativa, en este caso cumple con la función de un filtro que impide sean procesados los valores negativos del área, presentando un mensaje en la pantalla y dejando el programa preparado para que se le pueda entrar un nuevo valor del área.

```
10 REM (8)Programa para hallar el lado del cuadrado
20 INPUT "AREA=";A
30 REM Filtro de entrada
40 IF A<0 THEN 70
50 L=SQR(A)
60 PRINT L:STOP
70 PRINT "VALOR NEGATIVO":GOTO 20
80 END
```

Figura 29. Listado del programa n.º 8.

Esto lo podemos comprobar en el programa n.º 8 de la figura 29 que tiene indicado por el comentario REM la posición del filtro que impedirá el proceso de valores negativos. Al ponerlo en marcha por RUN queda en pantalla el mensaje «AREA=». Si le entramos el valor 2, queda como resultado el valor del lado 1,41421356, mientras que para valores negativos presenta el mensaje de «VALOR NEGATIVO».

Un detalle que hay que tener en cuenta en la programación es el STOP de la línea 60, el cual impide que permanezca en pantalla el mensaje del filtro después de dar el valor del lado. Podría haberse utilizado en su lugar GOTO 20 para el cálculo de nuevos valores. En el diagrama de flujo es fácil establecer ramas que van a sitios distintos, pero el programa es lineal y pueden procesarse zonas del programa que no están previstas.

LA PREPARACION PARA INICIALIZAR LA MARCHA DEL PROGRAMA

Una vez determinados los conceptos básicos de la

programación hay que detallar la forma de hacer que el computador funcione. En general se inicia con la orden RUN pero, a fin de que los resultados queden claros en la pantalla del computador, conviene borrarla generalmente con la orden CLS. La orden RUN deja también a cero las variables, pero si se quieren dejar a cero por programa puede utilizarse la orden CLEAR, esto puede ser interesante si no se utiliza RUN para poner en marcha el programa, sino otra orden como GOTO, que va a un paso determinado sin dejar a cero las variables.



Si el computador es capaz de desarrollar gráficos, conviene disponer de impresoras o copiadoras adecuadas, especialmente si los gráficos son en colores. (Cortesía: Tektronix).

Ya se ha indicado la utilidad de la orden REM para los comentarios de la programación, pero también suele usarse para poner título a los programas así como otros comentarios puesto que puede tener varias líneas, de modo que todas empiecen por REM formando como un recuadro en el listado y con toda clase de signos. Lo que va a continuación de esta orden no se procesará por el computador.

Antes de empezar un programa conviene eliminar el programa anterior que pueda estar en la memoria del computador, sobre todo si se trata de un modelo de los que tienen memoria continua. Normalmente, al desconectar el computador de la red eléctrica se borra todo su contenido,

pero esto no ocurre en los computadores que por ser portátiles funcionan con baterías de pilas o de acumuladores y entonces guardan permanentemente los datos en su memoria. Ya se ha indicado que las variables quedan a cero al empezar a funcionar el computador por RUN, pero el programa anterior se tiene que borrar por la orden NEW y después comprobar por LIST si está preparado para contener un nuevo programa.



Los computadores provistos de pantalla que admite varios colores resultan interesantes, sobre todo si han de representar gráficos o comparar diferentes variables. (Cortesía: Epson).

Cuando los programas se guardan en cassettes de audio se entran al computador con la orden CLOAD y se guardan en el cassette con CSAVE. Estas órdenes pueden llevar entre comillas el título del programa lo cual permite encontrarlas, aunque es poco práctica la búsqueda a la velocidad lenta del cassette, y como no puede funcionar a la velocidad rápida, se suelen encontrar fácilmente por el contador del cassette o habiendo grabado con palabras el título del programa. Es

lástima que no aparezcan en el mercado cintas cassettes que tengan el propio contador, lo cual facilitaría mucho el almacenamiento de programas en estas cintas. Actualmente hay que rebobinar la cinta hasta el origen, poner a cero el contador del aparato y luego buscar el programa por su numeración.

Generalmente se guarda con el programa el valor que tienen las variables en el momento de grabarlo. Esto puede ser muy interesante cuando se almacenan datos procesados, como por ejemplo resultados obtenidos o una gran lista de nombres de personas que constituyen la plantilla de una empresa. Como estos datos están a veces en lugares precisos y conocidos de la memoria del computador, pueden almacenarse y reclamarse bloques específicos de la memoria en donde se hallan los datos de las variables que pueden utilizarse para otros programas, por ejemplo.

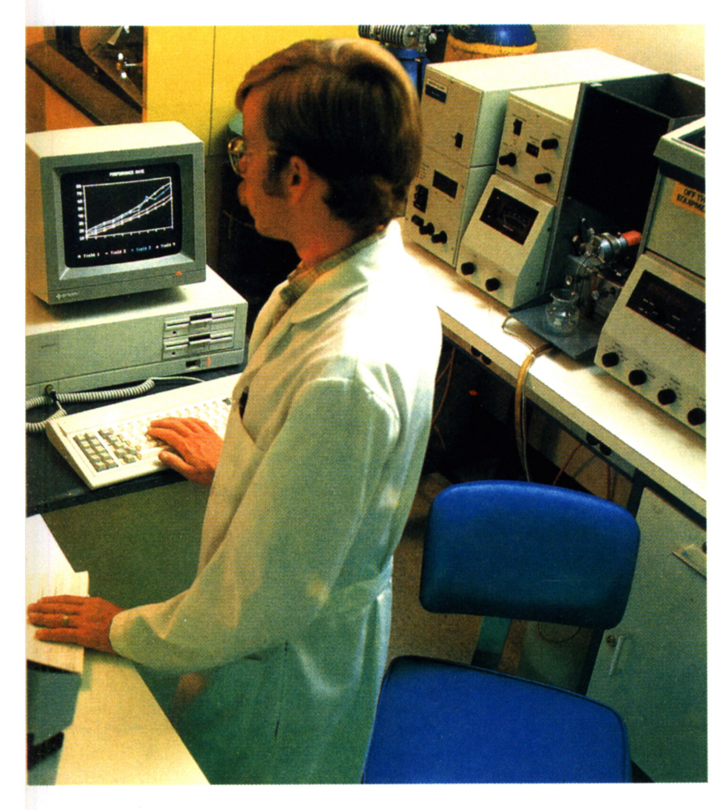
Finalmente queda por reseñar en este apartado que hay computadores que permiten que se ponga en marcha un programa una vez ha terminado de entrar de la cinta cassette, generalmente añadiendo la orden AUTO.

La entrada de datos

Supongamos que deseamos poner en marcha el programa y lo primero que debe hacerse es pedir unos determinados valores. Ya se ha indicado que este extremo se realiza con la orden INPUT pero no es la única forma de hacerlo en el lenguaje BASIC. También hay que decir que existen computadores que necesitan una detallada declaración de las variables que se van a utilizar así como su contenido.

En algunos casos hay que definir el valor que toman todas las variables, aunque sean nulas, pero más frecuente es tener que declarar la precisión que van a tener en su utilización. Entonces se utilizan órdenes como SHORT, INTEGER, etc., que dependen de cada computador. Una orden muy generalizada es DIM, que determina la dimensión de las variables con subíndice o el número de ellas. Si de A vamos a utilizar 25 variables distintas, es decir, la A(1), A(2), A(3), A(4)... A(25), hay que organizar internamente la memoria del computador poniendo al principio del programa la instrucción DIM A(25). Y lo mismo diríamos si se tratase de variables con dólar para almacenar cadenas de caracteres.

Sin embargo, hay computadores que ya tienen previamente dimensionadas las variables con 10 subíndices, es decir, que tendremos para las 26 letras 260 variables para asignar en el programa, sin necesidad de declararlas al principio de éste con la correspondiente comodidad en la programación.



Aplicaciones tan complejas como el análisis de productos, pueden facilitarse si se emplean computadores con los programas adecuados. (Cortesía: Sperry).

Veamos ejemplos de este dimensionamiento. Supongamos que deseamos utilizar un tablero de 6 × 6 casillas, las cuales deben contener unos valores determinados que entraremos al comenzar el programa. Estos valores pueden necesitarse, por ejemplo, para determinar el contenido de un tablero de un juego de estrategia, como el OTHELLO, en el que los valores indican la importancia de cada casilla, o bien un tablero de ajedrez reducido, etc.

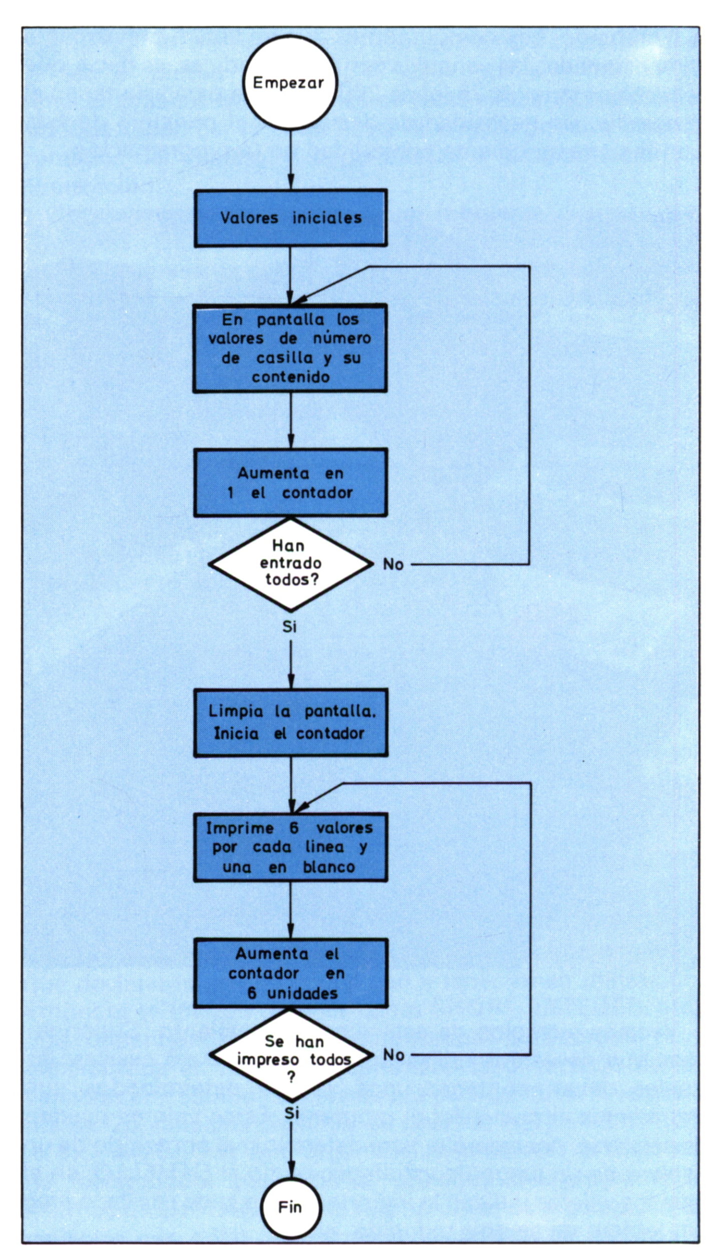


Figura 33. Diagrama de flujo correspondiente al programa n.º 9.

Podemos ver la forma de organizar esta entrada de datos en el diagrama de flujo de la figura 33, que corresponde al programa de la figura 34. Analizando su marcha observamos que hay una primera parte del programa que facilita la entrada de datos, mientras que la segunda parte imprime el resultado para que se pueda comprobar si se han llenado correctamente las casillas.

```
10 REM (9)Datos tablero 6x6
20 DIM A(36):N=1
30 REM Entran los valores en las casillas
40 PRINT N:INPUT A(N)
50 N=N+1
60 IF N<>37 THEN 40
70 REM Presentacion del tablero
80 CLS:N=1
90 PRINT A(N);" ";A(N+1);" ";A(N+2);" ";
100 PRINT A(N+3);" ";A(N+4);" ";A(N+5)
105 PRINT
110 N=N+6:IF N<>37 THEN 90
120 END


1  2  3  4  5  6

7  8  9  1  2  3

4  5  6  7  8  9

1  2  3  4  5  6

7  8  9  1  2  3

4  5  6  7  8  9
```

Figura 34. Listado del programa n.º 9.

En la línea 20 se encuentra el dimensionamiento inicial de la variable A y a continuación el valor inicial del contador N. Los distintos valores entran en la línea 40 con presentación en la pantalla por PRINT N del número de la casilla que está manejando, cuestión importante si el número de casillas fuese elevado, pues de lo contrario en pantalla aparece solamente la señal de entrar datos con una interrogación. A continuación, en la línea 50 aumentamos el valor del contador N en una unidad y en la línea 60 comprobamos si

se ha llegado al final de la entrada de datos, poniendo la condición de que si N es distinto de 37 se vuelve a entrar nuevo dato, y si ha llegado el contador al valor 37 se pasa a imprimir el resultado.

La impresión del resultado se hace por la línea 80, limpiando previamente todo lo que pudiese tener por CLS, y se vuelve a poner el contador a su valor inicial. Esta impresión se hace en grupos de 6 variables que figuran en el programa en dos líneas para que quede mejor su presentación. Puede observarse que el punto y coma de PRINT hace que quede en la misma línea, incluso al ponerlo en las dos líneas 90 y 100, pero que la última variable no tiene este punto y coma para que cambie de línea. También puede comprobarse que en la línea 105 se ha puesto la orden PRINT, sin ninguna variable a continuación para que quede una línea en blanco entre cada línea impresa y así queda el tablero más proporcionado.

El bucle de impresión del resultado aumenta cada vez en seis unidades en la línea 110, y si ya se ha llegado al final se detiene por END a través de la bifurcación condicional IF.... THEN....

Posteriormente veremos que las variables en forma de tabla de dos entradas pueden tener dos subíndices pero, por su sencillez, en este programa se ha preferido utilizar la variable dimensionada en la línea 20. Esta búsqueda de la sencillez hace que no se haya empleado el bucle repetitivo FOR... NEXT...., que verémos posteriormente.

Vamos a ver la forma de entrar datos a un programa, datos que están ya contenidos en el propio programa. Una forma podría ser utilizar los propios pasos del programa para poner las asignaciones de las variables, por ejemplo:

A(1) = 1:A(2) = 5:A(3) = 8:.... A(36) = 7, pero tiene el inconveniente de utilizar muchas veces la variable A lo cual gasta memoria inútilmente. Para evitar este inconveniente el lenguaje BASIC tiene la orden READ.... DATA.... que utilizaremos a continuación.

Cuando el programa encuentra READ, lee el valor de la lista DATA que tiene en el programa y lo asigna a la variable de READ.

Las listas de valores DATA pueden estar en cualquier lugar del programa, pero los datos han de estar separados por comas y se leerán secuencialmente, una después de otra, cada vez que aparezca READ y su variable. Si sobran valores

de DATA el programa los ignora, pero si faltan se detiene con un adecuado mensaje de error.

Por si hay que volver a leer los datos en otra parte del programa, se utiliza la orden RESTORE que vuelve a poner el índice de lectura de DATA a su origen.

```
10 REM(10)Datos tablero 6x6.-DATA
20 DIM A(36):N=1
30 REM Entrada de los valores
40 READ A(N):N=N+1
50 IF N<>37 THEN 40
60 REM Presentacion del tablero
70 CLS:N=1
80 PRINT A(N);"  ";A(N+1);"  ";A(N+2);"  ";
90 PRINT A(N+3);"  ";A(N+4);"  ";A(N+5)
100 PRINT
110 N=N+6:IF N<>37 THEN 80
120 DATA 4,7,1,4,7,1,5,8,2,5,8,2,6,9,3
130 DATA 6,9,3,7,1,4,7,1,4,8,2,5,8,2,5
140 DATA 9,3,6,9,3,6,7,7,7,7,7,7
150 END


4  7  1  4  7  1

5  8  2  5  8  2

6  9  3  6  9  3

7  1  4  7  1  4

8  2  5  8  2  5

9  3  6  9  3  6
```

Figura 35. Listado del programa n.º 10.

Esta forma de entrar datos a un programa la podemos ver en la figura 35 del listado del programa n.º 10. Puede observarse que se trata de una modificación del programa anterior, añadiéndole los datos de los valores que deseamos figuren en las casillas del tablero localizados en las líneas 120, 130 y 140, datos que son leídos por el READ de la línea 40 y asignados a la variable A(N). Puede comprobarse que los valores 7 de la línea 140 están en exceso y no alteran la marcha del programa, tal como se ha indicado.

Detenciones de la marcha del programa

Ya se han indicado las detenciones por STOP y la final por END, pero se necesita además interrumpir la marcha del programa cuando lo desee el usuario del computador, lo cual se realiza por la orden BREAK generalmente asociada a una determinada tecla o control. Mas si la detención ha de ser momentánea está la orden WAIT que suele ir seguida de un número que indica el tiempo de detención, normalmente en centésimas de segundo. Es muy útil para presentar mensajes en la pantalla del computador, mensajes que permiten comprobar el camino que está siguiendo y con ello desaparece la incertidumbre de la espera que puede necesitar un proceso de datos complicado. Piénsese lo que significa estar esperando unas horas el resultado del proceso de unos datos y cuando llega al final puede existir la duda de si el programa ha seguido el camino previsto o, lo que es peor, puede suceder que el programa entre por error de programación en un bucle continuo y estemos esperando que acabe inútilmente.



Computador personal de la firma Epson. Va provisto de una pantalla abatible, que facilita el transporte, con visualizador de cristal líquido.

Pueden utilizarse a este fin señales sonoras que generalmente se ejecutan con la orden BEEP o con otras, si el computador tiene posibilidades musicales.

Las bifurcaciones del programa

En programas anteriores se ha utilizado la orden GOTO para mandar el camino que sigue la programación a una línea determinada. Ello ha permitido en el programa anterior hacer la entrada y la impresión de los datos con un contador del tipo N = N + 1 y con una comparación para que se detenga al llegar a un valor dado. Todo lo citado anteriormente puede realizarse con una *orden para bucles repetitivos,* denominada FOR....NEXT.

```
10 REM (11)Datos tablero 6x6.FOR/NEXT
20 DIM A(36)
30 FOR N=1 TO 36
40 READ A(N):NEXT N
50 REM Presentacion del tablero
60 CLS
70 FOR L=0 TO 30 STEP 6
80 FOR N=1 TO 6
90 PRINT A(L+N);" ";
100 NEXT N
110 PRINT:PRINT:NEXT L
120 DATA 4,7,1,4,7,1,5,8,2,5,8,2,6,9,3
130 DATA 6,9,3,7,1,4,7,1,4,8,2,5,8,2,5
140 DATA 9,3,6,9,3,6
150 END

4  7  1  4  7  1

5  8  2  5  8  2

6  9  3  6  9  3

7  1  4  7  1  4

8  2  5  8  2  5

9  3  6  9  3  6
```

Figura 37. Listado del programa n.º 11.

El empleo de esta orden facilita grandemente la repetición de un trozo del programa porque se añade al término FOR una variable de control que va de un valor inicial hasta (TO) un valor final, con incrementos que se indican por un valor que sigue al término STEP. El trozo de programa a repetir termina con NEXT seguido de la misma variable de control.

Si no figura el término STEP el programa se realiza con incrementos de la variable de control de la unidad, pero estos incrementos pueden ser valores negativos con lo que el bucle disminuye de valor en la variable de control cada vez que se va repitiendo. Esta variable de control puede utilizarse dentro del bucle para asignarla a otras variables, pero ella no puede cambiar de valor si no es por su propio contador, que a la vez comprueba cada vez si ha llegado al valor indicado después del término TO.

Los bucles FOR-NEXT pueden incluirse unos dentro de otros según un número de bucles que se especifican para cada computador, ya que deben tener una memoria que retiene los bucles pendientes de terminar por NEXT. Si hay comparaciones del tipo IF....THEN... hay que procurar no entrar en un bucle por un paso de programa sin haber pasado antes por FOR.... TO...., de lo contrario aparecerá un mensaje de error en la pantalla.

```
10 REM (12)Diagonal maxima
20 INPUT "LARGO=";L
30 INPUT "ANCHO=";A
40 INPUT "ALTO=";B
50 C=SQR(L*L+A*A)
60 D=SQR(C*C+B*B)
70 PRINT D
80 END
```

Figura 38. Listado del programa n.º 12.

Una aplicación de estos bucles repetitivos podemos verla en el programa n.º 11 de la figura 37, que es el programa anterior con la aplicación de tres bucles repetitivos. En la línea 30 está el bucle que permite entrar los 36 valores que están en DATA. La variable de control es N, que tomará los valores desde 1 a 36 teniendo el NEXT en la línea 40. Para la impresión del resultado se han incluido dos bucles, uno dentro de otro.

Uno de ellos tiene por variable de control L, y como cada vez aumenta en 6 unidades (STEP) representa cada línea del tablero, mientras que el otro bucle tiene por variable de control N y representa cada uno de los valores de cada línea.

Para que deje una línea en blanco entre dos líneas hay que usar dos veces la orden PRINT, puesto que uno de ellos tiene que anular el efecto del punto y coma de la línea 90 y el otro deja la línea en blanco.

La posición de cada casilla viene determinada por la suma de las dos variables de control, tal como figura en el subíndice de la variable A de la línea 90. Por lo demás, el diagrama de flujo sería el mismo de la figura 33, con supresión de los contadores, que ahora ya no hay que programar, y las bifurcaciones condicionales.



La importancia de los programas es decisiva cuando los computadores están dedicacos al diseño industrial. (Cortesía: Hewlett Packard).

Otro tipo de bifurcaciones del programa parecidas al GOTO son las que utilizan *subrutinas*. Una *subrutina* es un trozo de programa que se ejecuta las veces que haga falta y termina siempre con la orden RETURN para que el camino que sigue el programa retorne a la línea siguiente, la cual había abandonado para ir a la subrutina. Para ir a ella se

utiliza la orden GOSUB seguida del número de la línea del programa en donde empieza la subrutina.

Con un ejemplo podemos comprobar la utilidad de esta bifurcación. Supongamos que estamos en una habitación de

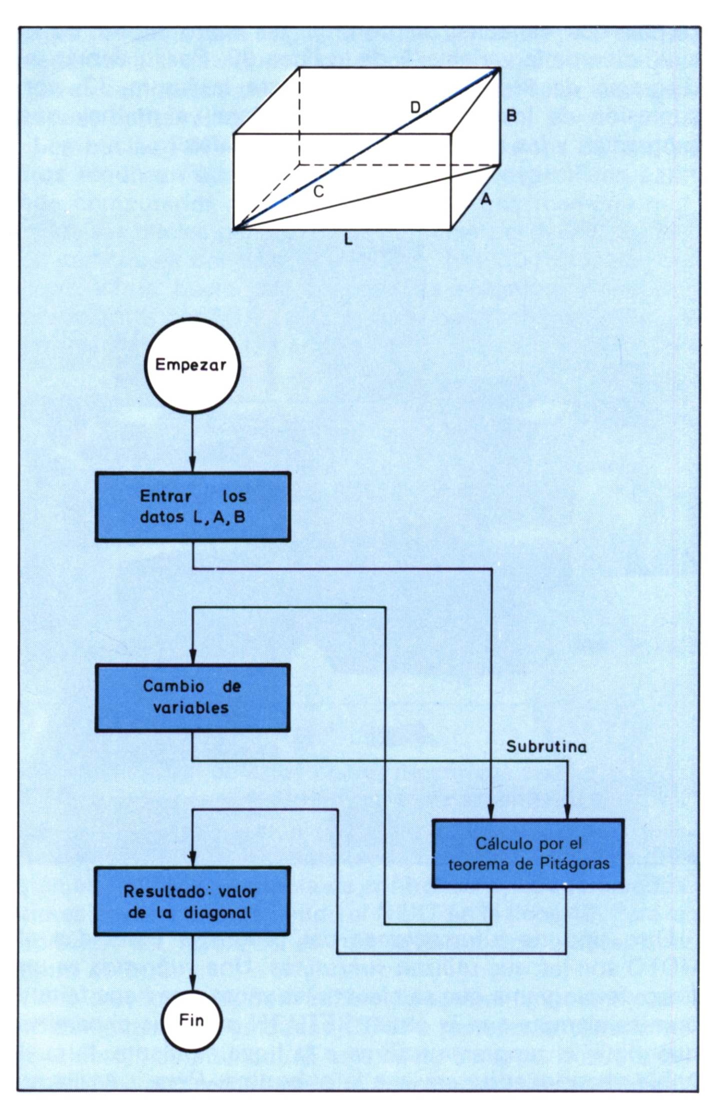


Figura 40. Diagrama de flujo del programa n.º 13.

paredes rectangulares y queremos saber el valor de la diagonal que hay entre dos ángulos opuestos. Para ello tenemos las medidas de las tres dimensiones: largo = L, ancho = A, y alto = B. El valor de la diagonal lo encontraremos haciendo dos veces el cálculo de la hipotenusa por el Teorema de Pitágoras. Este será el algoritmo que usaremos y para ello un primer programa puede ser el n.º 12 de la figura 38 que apenas presenta dificultad por reducirse a entrar los datos y luego calcular dos veces la diagonal en las líneas 50 y 60.

Pero el hecho de que tenga que calcularse dos veces la diagonal por el Teorema de Pitágoras permite poner este cálculo como una subrutina del programa, aunque sea una subrutina corta.

Esto lo podemos ver en la figura 40 que representa el programa con subrutina, programa cuyo listado está en la figura 41 y con el n.º 13. La primera bifurcación a subrutina está en la línea 50 y, después de calcular la primera diagonal, al volver a la línea 60 tiene que hacerse un cambio de variables para poder emplear la subrutina una segunda vez en la línea 70.

```
10 REM (13)Diagonal maxima.GOSUB
20 INPUT "LARGO=";L
30 INPUT "ANCHO=";A
40 INPUT "ALTO=";B
50 GOSUB 100
60 L=C:A=B
70 GOSUB 100
80 PRINT C
90 STOP
100 C=SQR(L*L+A*A):RETURN
110 END
```

Figura 41. Listado del programa n.º 13.

Es importante el STOP de la línea 90, ya que de lo contrario el programa efectuaría los cálculos una tercera vez y como encontraría un RETURN sin tener antes un GOSUB, en la pantalla aparecería un mensaje de error por este motivo. Puede verse también que se ha preferido poner los

cuadrados de los lados en forma de producto para tener mayor exactitud que empleando la potenciación.

Esto puede comprobarse con los datos de L = 72, A = 16, B = 63 que debe dar un resultado del valor de la diagonal de 97 exacto.

Dentro del apartado de las bifurcaciones que puede seguir la marcha del programa hay que considerar la orden ON..... GOTO, que también admite ON.... GOSUB. Con ella se manda a un número de líneas que depende del valor que tiene una variable de control que sigue al término ON. Si el valor de esta variable es 1, el programa irá al número de línea que está en primer lugar después de GOTO, si vale 2 al segundo, y así sucesivamente. Las distintas líneas tienen que existir en el programa, de lo contrario aparecerá un mensaje de error con la consiguiente detención.

Como ejemplo de utilización de esta versátil orden se presenta en el programa n.º 14 de la figura 42 el siguiente caso para programar. Un generador aleatorio nos entrega números enteros comprendidos entre 1 y 4. Deseamos saber si después de generar cien números ha aparecido igual cantidad de 1 que de 2, de 3 ó de 4, lo cual nos da una cierta confianza en su bondad.

En el programa figuran por REM los comentarios de su funcionamiento. Así, en la línea 20 figura la fórmula del generador aleatorio que entrega números decimales, comprendidos entre 0 y 1, sin alcanzar nunca este último valor. En la línea 40 se acotan los números generados entre 1 y 4, para bifurcar en la línea 60 a las distintas líneas del programa que acumulan las veces que ha salido cada número, veces que se acumulan a las variables C, D, E, F. El contador de la línea 510 y la comparación de si ha llegado a generar cien números, establece la bifurcación condicional para seguir generando números o bien imprimir el resultado. También aquí podría haberse utilizado un bucle repetitivo FOR... NEXT, en lugar del contador y la comparación.

Por el resultado de los cien números generados se observa que aparecen muchos menos del número 2, pero seguramente hay que comprobar si generando más números cambia este hecho y a la vez hay que tener en cuenta que se ha partido de que el valor A empieza a cero y debería modificarse el programa para poder empezar por cualquier número.

Antes de terminar este apartado hay que citar las

posibilidades que tiene la bifurcación condicional por la orden IF... THEN.... ELSE. Ya se ha usado la bifurcación condicional en otros programas, pero el hecho de que si no se cumple la condición se pase a la línea siguiente hace que esta orden pueda quedar mejorada con la adición del término ELSE, que por otra parte no figura en todos los computadores.

```
10 REM (14)Frecuencia generacion al azar
20 REM Generador aleatorio
30 A=(A+PI)^5-INT((A+PI)^5)
40 B=INT(4*A+1)
50 REM Bifurcacion
60 ON B GOTO 100,200,300,400
100 REM veces num.1
110 C=C+1:GOTO 500
200 REM veces num.2
210 D=D+1:GOTO 500
300 REM veces num.3
310 E=E+1:GOTO 500
400 REM veces num.4
410 F=F+1
500 REM contador y resultado
510 N=N+1:IF N<100  THEN 30
520 PRINT"Veces num.1=";C
530 PRINT"Veces num.2=";D
540 PRINT"Veces num.3=";E
550 PRINT"Veces num.4=";F
560 END


Veces num.1=36
Veces num.2=14
Veces num.3=22
Veces num.4=28
```

Figura 42. Listado del programa n.º 14.

Ya se ha indicado que la bifurcación condicional es cómoda de interpretar en el diagrama de flujo pero que en el programa, debido a que es lineal, si no se cumple la condición IF se pasa a la línea siguiente. Este inconveniente queda mejorado al utilizar ELSE, como podremos ver en el programa n.º 15 de la figura 43, cuyo diagrama de flujo está en la figura 44.

En vez de ir a un número de línea con la bifurcación condicional, se prefiere en este caso utilizar la orden PRINT

para poder mostrar otra de las posibilidades que ofrece el lenguaje BASIC. El programa sirve para adivinar un número generado como antes al azar y comprendido entre 1 y 4. Si se suprime la línea 35 se podrá adivinar el número generado, pero si se deja se podrá comprobar la forma en que responde el computador por la pantalla.

```
10 REM (15)Acertar un numero
20 A=(A+PI)^5-INT((A+PI)^5)
30 B=INT(4*A+1)
35 PRINT B
40 INPUT "QUE NUMERO";N
50 REM Bifurcacion
60 IF N>B THEN PRINT"ES MAYOR":GOTO 20
70 IF N<B THEN PRINT"ES MENOR"ELSE PRINT"EXACTO"
80 GOTO 20
90 END
```

Figura 43. Listado del programa n.º 15.

Como podemos ver en el diagrama de flujo la condición tiene tres alternativas: El intento de adivinar el resultado con un número mayor, o bien con un número menor, o si se ha dado un resultado exacto. Para ello utilizamos dos veces la orden IF... THEN y en la segunda por ELSE tenemos el resultado exacto.

La facilidad de utilizar PRINT en la bifurcación condicional es otra más de las posibilidades que, además de ir a un número de programa, permite establecer una asignación entre variables por complicadas que sean. Queda además para reseñar en el tema de las bifurcaciones del programa que se establecen de modo condicional, las que emplean los operadores booleanos AND, OR, NOT, EXOR, etc.; estas se intercalan entre el IF y el THEN y permiten concretar al máximo las condiciones de las bifurcaciones del programa.

EL PROCESO DE LOS DATOS

Después de que se han visto los variados caminos que puede seguir el programa, hay que considerar la forma en que procesa los datos. En esencia se trata de calcular, pero este cálculo también puede ser comparar, asignar, anular,

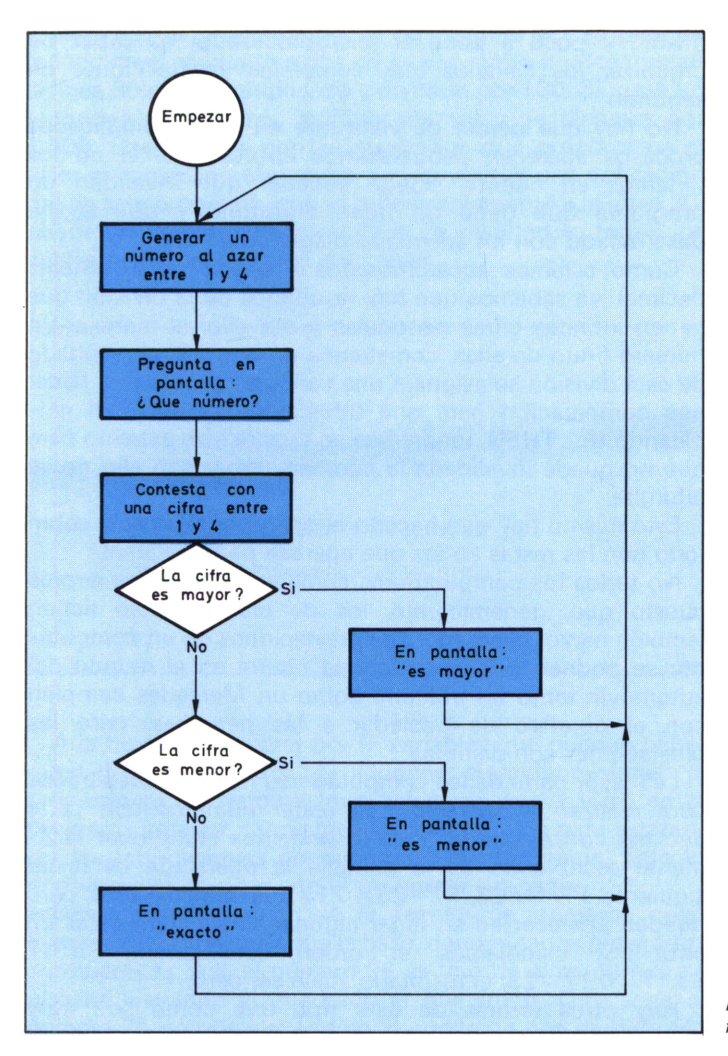


Figura 44. Diagrama de flujo del programa n.º 15.

etc. La parte más importante del computador es el microprocesador, que organiza todo el trabajo, pero en realidad está preparado para hacer pocas cosas: *sumar* y *comparar*. A pesar de ello, como lo hace a una velocidad rapidísima, por programa la suma pasa a multiplicación, por tomar el inverso ya se tiene la resta, la cual fácilmente se programa para

dividir, y poco a poco el microprocesador es capaz de organizar los cálculos más complejos en fracciones de segundo.

No hay que perder de vista que en estos complicados procesos aparecen pequeñísimos errores que, si no los tenemos en cuenta, puede suceder que invaliden un programa que tiene un buen algoritmo y que se ha desarrollado con un adecuado diagrama de flujo.

Como estamos acostumbrados a operar en el sistema decimal, ya sabemos que hay resultados de la división que tienen infinitas cifras periódicas y por ello, al manejar un número finito de ellas, cometemos un error. Si el resultado de esta división se asigna a una variable numérica, al hacer una comparación para una bifurcación condicional empleando IF... THEN, tendremos en cuenta este extremo para que no quede invalidada la comparación y con ello no se bifurque.

Esto mismo hay que hacerlo al operar con raíces y sobre todo con las restas en las que aparece parte decimal.

No todos los computadores cometen los mismos errores puesto que, generalmente, los de mayor precio tienen también mayor exactitud. Las prestaciones de un computador se podrían asimilar a lo que ocurre en el mundo del automóvil: tanto un utilitario como un Mercedes cumplen con el objetivo de trasladar a las personas, pero las prestaciones son distintas.

La mayor parte de los computadores no están preparados para realizar la operación de restar números con parte decimal con exactitud. Esto lo podemos comprobar fácilmente escribiendo en la pantalla la operación de restar siguiente: PRINT 23, 17 – 23 – 0,17 que debe dar cero, pero pueden aparecer en su lugar algunas cienmillonésimas de error. Si cambiamos el orden escribiendo: PRINT 23,17 – 0,17 – 23, el resultado suele ser cero.

Hay otros errores de este tipo que como son muy pequeños no se notan en el resultado de la pantalla, pero otra cosa es emplearlos en los bucles condicionales en los que la condición, al no ser verdadera, pasará el programa a la línea siguiente y nosotros esperábamos que seguiría en la misma línea. Por ello hay que ver como se pueden corregir.

Los pequeños errores en el proceso de los datos numéricos se suprimen habitualmente empleando un redondeo a un número de cifras dado. Para ello podemos ver la forma de

realizarlo en el programa n.º 16 de la figura 45. Se trata de redondear el resultado a dos cifras decimales puesto que en la línea 30 se ha multiplicado y dividido por 100. Si fuese un redondeo a tres cifras, multiplicaríamos y dividiríamos por 1.000. Puede verse por los resultados que proporciona el valor más cercano al número a redondear y una aplicación útil de este programa, que se reduce solamente a la línea 30, puede ser el confeccionar tablas de resultados, todos los cuales tienen los mismos decimales.

```
10 REM (16)Redondeo a centesimas
20 INPUT "NUM.A REDONDEAR=";A
30 A=INT(A*100+0.5)/100
40 PRINT A
50 GOTO 20

NUM.A REDONDEAR=3.14159265
3.14
NUM.A REDONDEAR=2.71828182
2.72
```

Figura 45. Listado del programa n.º 16.

A la hora de comparar por IF variables que pueden haber quedado con el error de algunas cienmillonésimas, si previamente las redondeamos tal como se ha visto en el programa anterior, tendremos valores seguros para establecer las bifurcaciones condicionales del programa. Esto lo podremos comprobar con el programa siguiente.

Deseamos investigar en la familia de números cuyo cuadrado termina igual que la base, denominados *cuadrados automórficos*. Así, el número 36 es uno de ellos porque se obtiene elevando el cuadrado el número 6. Otro sería el número 25 porque elevándolo al cuadrado nos da 625. No sabemos cuántos números tienen esta propiedad y establecemos el algoritmo de encontrarlos, buscándolos de forma consecutiva y en cada uno hallamos el cuadrado y comprobamos si termina con las mismas cifras que la base. Como durante la búsqueda irá aumentando el número de sus cifras, el cuadrado tendrá que compararse cada vez con un número de cifras adecuado, de manera que si el número no llega a 10

se comprobará una sola cifra terminal del cuadrado, si no llega a 100 se comprobarán dos cifras, y así sucesivamente. La comprobación se hará dividiendo el cuadrado por 10,

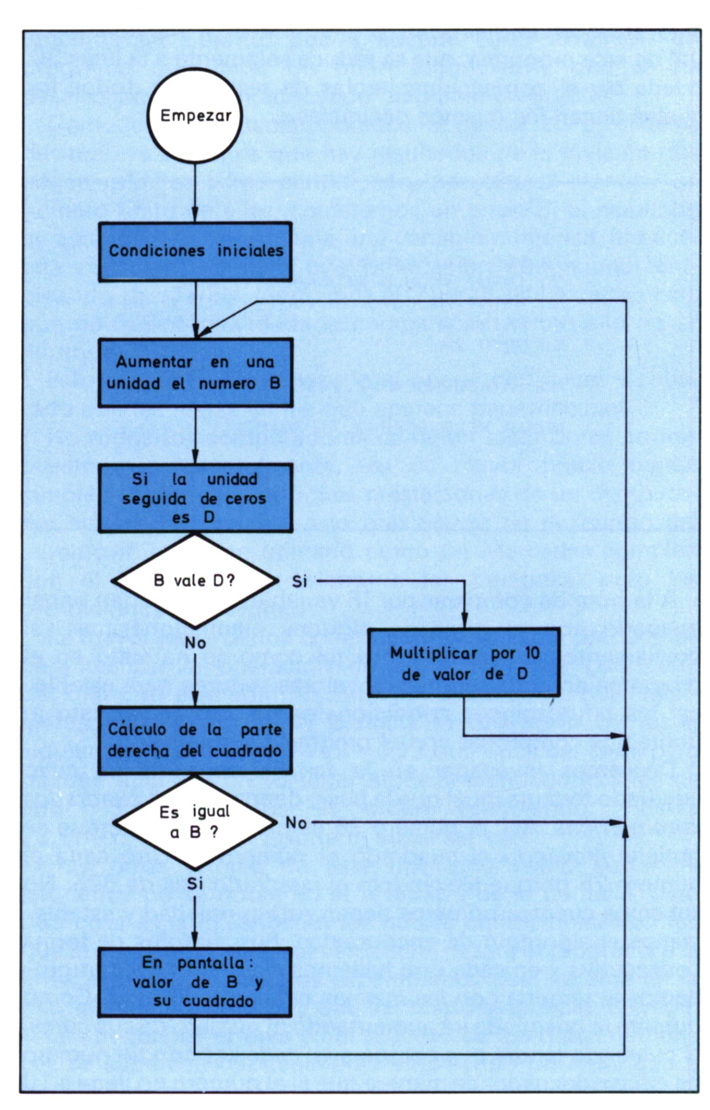


Figura 46. Diagrama de flujo del programa n.º 17.

100, etc., tomando la parte decimal y luego multiplicándola por 10, 100, etc. Así, el número 76 elevado al cuadrado da 5.776, que dividido por 100 queda 57,76. La parte decimal 0,76 multiplicada por 100 nos da 76, que coincide con la base.

Este algoritmo lo traducimos al correspondiente diagrama de flujo que está en la figura 46 y luego confeccionamos el programa n.º 17, de la figura 47. Puede observarse el redondeo de la línea n.º 35, debido a que determinados computadores procesan la línea 30, en la que está la resta, para tener la parte decimal empleando la función INT con pequeños errores, los cuales no permitirían la bifurcación condicional IF F = B de la línea 40.

```
5 REM (17)Cuadrados automorficos
10 B=0:D=10
20 B=B+1:IF B=D THEN 80
30 C=B*B:E=C/D-INT(C/D)
35 E=INT(E*10000+.5)/10000
40 F=INT(E*D):IF F=B THEN 60
50 GOTO 20
60 PRINT "NUM.=";B;" CUADR.=";C
70 GOTO 20
80 D=10*D:GOTO 20
90 END

NUM.=1   CUADR.=1
NUM.=5   CUADR.=25
NUM.=6   CUADR.=36
NUM.=25   CUADR.=625
NUM.=76   CUADR.=5776
NUM.=376   CUADR.=141376
NUM.=625   CUADR.=390625
```

Figura 47. Listado del programa n.º 17.

Hay computadores que tienen además de la función INT (parte entera), la función FRAC (parte decimal o fraccionaria) de un número. Cuando no se dispone de esta función puede suplirse restando todo el número de su parte entera, y es en esta resta donde aparecen los errores. Si se suprime esta línea 35, algunos números no aparecen en el resultado obtenido por el computador, como el 6 y el 376, aunque depende del tipo de computador. Puede comprobarse que el

programa tiene en la línea 20 el contador que hace aumentar los números a ensayar de uno en uno. En la línea 80 se tiene el aumento de una cifra de la parte final del cuadrado, porque el número pasa por ejemplo de 99 a 100, cuestión que se bifurca en la línea 20.

La programación de esta familia de números demuestra que terminan en 5 o en 6, a excepción del número 1, y que es una familia poco numerosa, puesto que después del número 625 no se encuentra otro hasta el 9376, lo cual hace que se tenga que esperar mucho a medida que se sigue buscando nuevos números, pero esto puede no significar gran cosa si es posible dejar el programa funcionando horas y horas, como por ejemplo por la noche.

Otra forma de evitar los errores del proceso de datos es la de usar la multiplicación en vez de la potenciación, que ya se ha usado en el programa para obtener el valor de la diagonal máxima de una habitación, pero lo normal es usar la programación para tener valores en los que la precisión no cuenta.

El siguiente programa es otra de las posibilidades del lenguaje BASIC. Se trata de realizar con un programa el cálculo de las calorías que debe ingresar una persona con la comida, procesando fórmulas complejas, pero por la pantalla del computador podemos tener una gran ayuda al presentar con los respectivos mensajes los datos que hay que introducir en las fórmulas.

Esto podemos verlo en el diagrama de flujo de la figura 48, que corresponde al programa n.º 18 de la figura 49. Se parte de unos coeficientes que se asignan a variables para que las fórmulas sean más cortas y luego se pregunta por los datos personales de peso, estatura y edad. Como el metabolismo tiene distinta fórmula en el hombre que en la mujer, este extremo aparece en la pantalla con las iniciales que hay que pulsar, las cuales cambian el camino de la programación por la bifurcación condicional de la línea 100. Puede observarse que se comparan letras, por lo que en la variable X se añade el signo dólar.

Después del primer cálculo que proporciona el metabolismo basal o las calorías que se consumen al dormir en cada hora, se pasa a entrar las horas que se emplean en cuatro tipos de trabajo distintos. El *ligero* es un trabajo sedentario como escribir, coser, comer, etc. El *medio* es un trabajo que significa estar de pie, andar sosegadamente, trasladarse, etc.

El trabajo *fuerte* se realiza en cualquier actividad u oficio manual, como cavar, pintar una pared, etc., mientras que el trabajo *muy fuerte* se puede aguantar poco tiempo porque es del tipo de correr, hacer deporte activamente, etc.

Una vez entrados estos tipos de trabajo a sus variables, se comprueba por la línea 410 si el número de horas da 24, porque de lo contrario aparece el mensaje de error en el número de horas y se vuelven a entrar las que corresponden a cada tipo de trabajo. Se procesan estas horas por unos

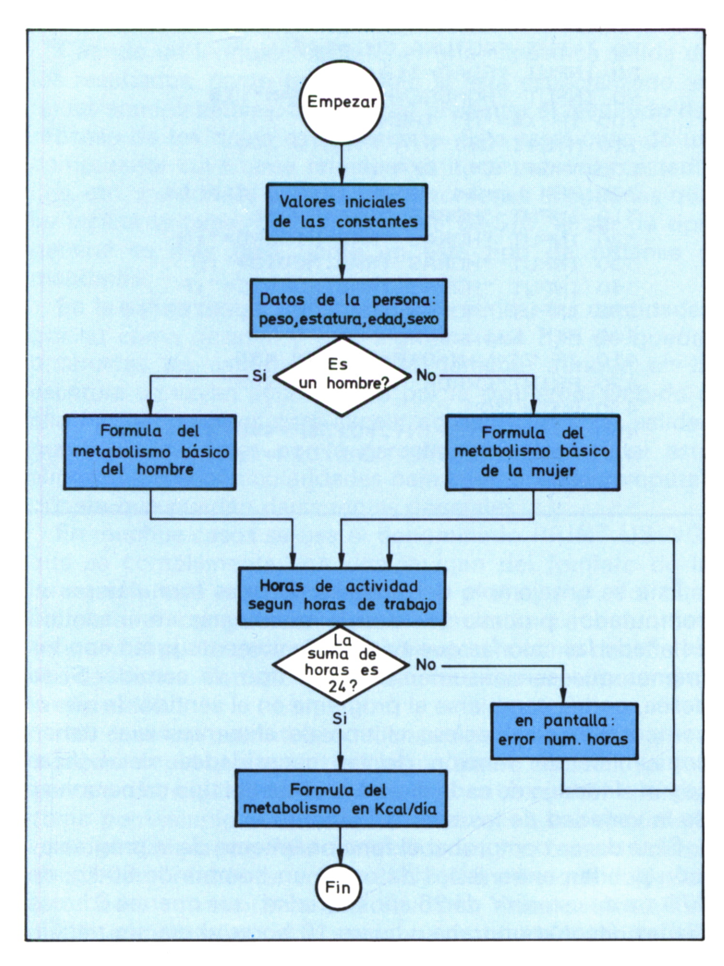


Figura 48. Diagrama de flujo del programa con el que se realiza el cálculo de las calorías que debe ingerir diariamente una persona.

coeficientes, quedando finalmente en pantalla el resultado en kilocalorías por día, valor que a veces se denomina calorías simplemente.

```
10 REM (18)Kilocalorias diarias
20 REM Constantes de la formula
30 C=.425:D=.725:E=.504:F=.167
40 G=.442:H=.151
50 REM Datos de la persona
60 INPUT "PESO KG.=";I
70 INPUT "ALTURA CM.=";J
80 INPUT "EDAD=";A
90 INPUT "HOMBRE=H,MUJER=M";X$
100 IF X$="M" THEN 200
110 M=E*I^C*J^D/A^F:GOTO 300
200 M=G*I^C*J^D/A^H
300 REM Horas de actividad diaria
310 INPUT "HORAS DORMIR=";L
320 INPUT "HORAS TRAB.LIGERO=";N
330 INPUT "HORAS TRAB.MEDIO=";O
340 INPUT "HORAS TRAB.FUERTE=";P
350 INPUT "HORAS TRAB.MUY FUERTE=";Q
400 REM Calculos
410 IF 24=L+N+O+P+Q THEN 450
420 PRINT"ERROR NUMERO HORAS"
430 GOTO 300
450 R=INT(L*M+(N+2*O+3*P+4*Q)*I*1.154)
460 PRINT "TOTAL KCAL/DIA=";R
470 END
```

Figura 49. Listado del programa n.º 18, correspondiente al ejercicio propuesto en la figura anterior.

Este es un ejemplo del proceso de unas fórmulas por el computador, proceso que podría modificarse en el sentido de añadir las calorías que tiene cada alimento junto con los gramos que se consumen en cada tipo de comida. Si se desea, podría cambiarse el programa en el sentido de que el computador estableciese el tipo de alimentos que deben consumirse en función de las necesidades metabólicas según el trabajo de cada día, en función del tipo de persona y de la variedad de los menús.

Si se desea comprobar el funcionamiento de la programación pueden entrarse los datos de un hombre de 60 kg, de 170 cm de estatura, de 25 años de edad, que duerme 8 horas diarias, efectúa un trabajo ligero 10 horas al día, un trabajo

medio 6 horas, un trabajo fuerte de 0 horas así como uno muy fuerte de 0 horas, entonces el resultado debe dar 2.079 Kcal/día.

El apartado del proceso de los datos por el computador tiene que complementarse posteriormente con el tratamiento de las cadenas de caracteres, que amplía enormemente las posibilidades de los lenguajes de alto nivel.

LA PRESENTACION DEL RESULTADO

Cuando un lenguaje debe tener una específica salida de los resultados, como por ejemplo si está especializado en tareas administrativas, la forma de presentar el resultado del proceso de los datos es importante. Este es el caso de un computador cuya tarea principal es hacer *nóminas*, estadillos, etc. y entonces se procura por órdenes adecuadas que se facilite la tarea. Pero el lenguaje BASIC, al ser de tipo general es más bien pobre en este tipo de órdenes o mandatos.

En la salida de los datos conviene alinear las cantidades por su coma decimal y esto significa que han de quedar ordenadas las cantidades por la derecha, aunque en la escritura se vayan acumulando por la izquierda. Debido a ello hay dificultades para alinear correctamente las salidas numéricas de datos por la derecha, y a fin facilitar esta alineación hay particularidades para cada tipo de computador sin que puedan darse reglas generales.

En muchos casos se usa el denominado PRINT USING, que se complementa con una imagen del formato de la impresión con símbolos como ####.##, lo cual indicaría al computador que el resultado tiene en una columna el punto decimal (equivalente a la coma decimal) y dos decimales. Con ello quedan las cantidades perfectamente alineadas.

Hay casos en que interesa rellenar el resultado con ceros a la izquierda sin que aparezca ningún punto decimal, ya que se trata de resultados de cifras que se van acumulando, como por ejemplo al obtener las cifras que tiene el número *«pi»* o el número *«e»*. Para dar facilidades, hay computadores que tienen órdenes parecidas al USING pero que dejan las cantidades con los ceros a la izquierda. Veremos posteriormente cómo realizar este caso por programa, cuando hay

que obtener las cifras del número «*e*» colocadas en 5 grupos de 5 cifras cada uno.

Otros computadores permiten obtener espacios en blanco en la pantalla con órdenes sencillas como el SPC (N) y en general es muy usado el PRINT TAB (N), que deja el resultado desplazado N columnas y las cifras alineadas por la izquierda, con lo que no quedan las comas alineadas en columna.

```
10 REM (19)Num.cifras variable
20 INPUT "SEMILLA=";N
30 GOSUB 300:A=N
40 A=INT(7*A+1):GOSUB 300
50 L=INT(N*10^A):B=B+1
60 REM Impresion de 4 numeros
70 LPRINT SPC(7-A);L;
80 IF B<>4 THEN 30
90 B=0:LPRINT:GOTO 30
300 REM Generador aleatorio
310 N=N*2939+PI-INT(N*2939+PI)
320 RETURN

 852644        2        9       28
    289  1709566    79670     8719
    907      901  7548770   979993
    483      170     6585     9620
   5153   909175       81       74
     60   387745   815692     9891
     43  6282691    30607    38021
  41703   147533   774354  1642055
  19771       63        9        7
9804039     6401     5415  2073097
    829     2409       10   475661
      6    16584     5259   239162
   1298    43657       41     1075
```

Figura 50. Listado del programa n.º 19.

Si se trata de colocar un resultado en el área de la pantalla por las coordenadas L (línea) y C (columna), hay computadores que tienen la orden PRINT AT L, C, que proporciona una gran riqueza de posibilidades. De todos modos, tanto el colocar espacios vacíos como el colocar ceros a la izquierda tiene que complementarse con el obtener por programa el número de cifras de la variable, caso que vamos a ver en

el programa siguiente, obtenido con la función LOG y en su parte entera.

El programa n.º 19 de la figura 50 obtiene cifras al azar

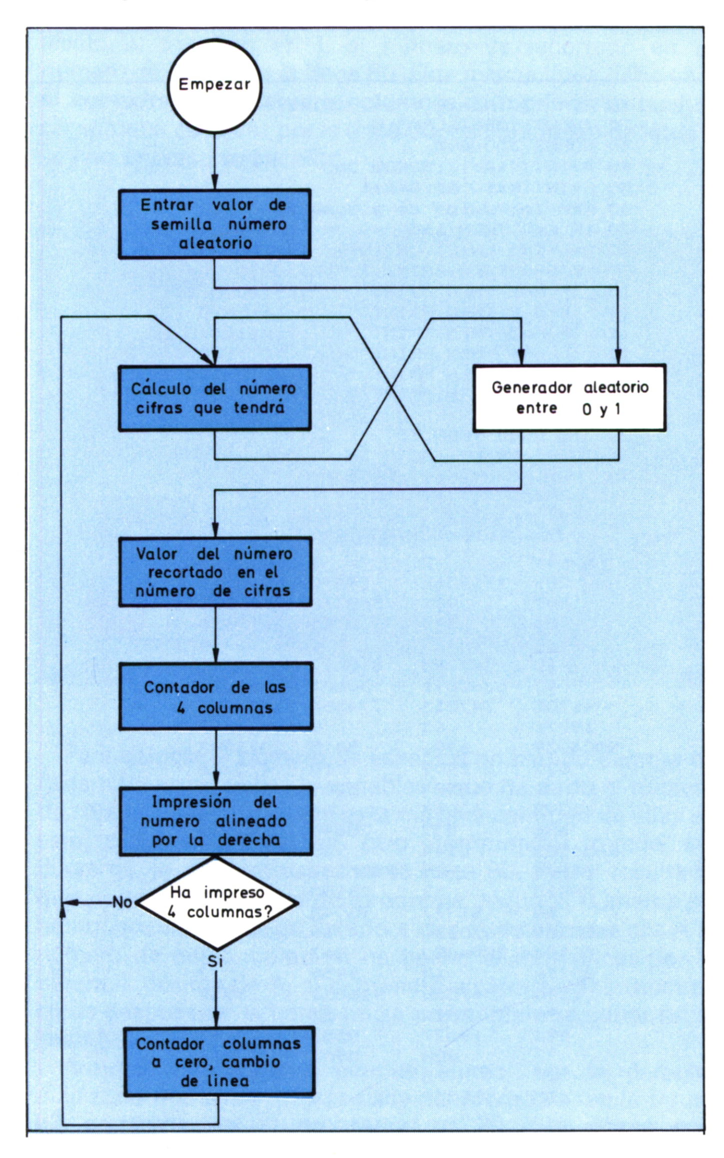


Figura 51. Diagrama de flujo del programa n.º 19.

con el generador indicado en la línea 300. A la vez, se desea que el total de las cifras de cada número sea también variable al azar, con lo que se ha de resolver el problema de alinearlas bien presentándolas en cuatro columnas.

```
10 REM (20)Num.cifras var.derecha
20 INPUT "SEMILLA=";N
30 GOSUB 300:A=N
40 A=INT(7*A+1):GOSUB 300
50 L=INT(N*10^A):B=B+1
60 REM Impresion de 4 numeros
70 IF L=0 THEN 160
80 G=6-INT(LN(L)/LN(10))
90 IF G=0 THEN PRINT " ";L;:GOTO 200
100 IF G=1 THEN PRINT "  ";L;:GOTO 200
110 IF G=2 THEN PRINT "   ";L;:GOTO 200
120 IF G=3 THEN PRINT "    ";L;:GOTO 200
130 IF G=4 THEN PRINT "     ";L;:GOTO 200
140 IF G=5 THEN PRINT "      ";L;:GOTO 200
150 IF G=6 THEN PRINT "       ";L;:GOTO 200
160 PRINT "       0 ";
200 IF B<>4 THEN 30
210 B=0: PRINT: GOTO 30
300 REM Generador aleatorio
310 N=N*2939+PI-INT(N*2939+PI)
320 RETURN
```

```
  852644        2        9       28
     289  1709566    79670     8719
     907      901  7548770   979993
     483      170     6585     9620
    5153   909175       81       74
      60   387745   815692     9891
      43  6282691    30607    38021
   41703   147533   774354  1642055
   19771       63        9        7
 9804039     6401     5415  2073097
     829     2409       10   475661
       6    16584     5259   239162
    1298    43657       41     1075
     686      532    28848    93972
 2651805     2144      933  4476203
   34063       25  9355592   260476
   26843        7    78190       62
      88    81777    73048        4
 9165506  9620180     4766  4200391
   50144  2294082      863       37
   88776       18     9959        2
    1240   547356   928031     9939
      98     5546       25       56
     493    19677     9583    72641
       7      888     5901      816
       7        8        4  1092929
     585   373239        0    76503
```

Figura 52. Listado del programa n.º 20.

En el diagrama de flujo de la figura 51 podemos ver que primero y por una subrutina obtenemos la cantidad de cifras que va a tener, valor que queda en la variable A en la línea 30 y 40. Utilizando por segunda vez la subrutina del generador aleatorio, tenemos en L el número ya recortado en su número de cifras por la línea 50. Esta misma línea tiene en B el contador de las cuatro columnas a imprimir, lo cual se comprueba cada vez por la línea 80, cambiándose de línea si se han impreso todas ellas.



Sistema X-07 de Canon. A pesar de su reducido tamaño, acepta diferentes tarjetas con programas y otros periféricos interesantes, como es el caso de un acoplador óptico que permite su conexión a otro computador más potente, un convertidor para adaptarlo a otros tipos de computadores y una impresora a color.

Para colocar el número de espacios en blanco delante de cada número se utiliza la variable A que ha dado el número de cifras al restar de 7, que es la máxima cantidad de ellas en este programa, pero en otro computador puede ser diferente, lo cual se realiza en la línea 90. Por el resultado que podemos ver al pie del programa, algunos números no han quedado alineados debido a que al determinar por A el número de cifras, como se ha hecho al multiplicar por la potencia décima de A, si el número aleatorio tiene muchos ceros después de la coma no se presentarán al cortar en la línea 50.

Aunque este sistema permite alinear por la derecha utilizando una parte muy sencilla del programa, en la figura 52 podemos ver el programa n.º 20 que alinea con



Los niños disfrutan tomándose el manejo de los computadores como un juego; más adelante, en su vida profesional, tendrán que emplearlos para desarrollar un gran número de trabajos. (Cortesía: Thomson).

seguridad las cantidades a base de medir el número de cifras con el logaritmo decimal e imprimir según varios bucles condicionales. El cálculo del número de cifras se realiza con el logaritmo decimal en la línea 80, utilizando el neperiano si la función LOG no está en el computador y dividiéndolo por el logaritmo de la base. Como en este cálculo no puede entrar el logaritmo de cero, en la línea 70 se elimina este caso enviándolo a la 160 que lo imprimirá correctamente.

Si el número de cifras es 7 se imprime el espacio de separación en la línea 90, si se trata de 6 cifras se imprimen dos espacios por la línea 100 y así sucesivamente, pudiéndose ver por el resultado su perfecta alineación, incluso cuando se obtiene al azar el valor cero que figura en la última línea del resultado. El resto del programa funciona como el anterior, comprobándose en la línea 200 si se han impreso las 4 columnas y, si es así, se pone el contador a cero y se pasa a la línea siguiente, todo ello por la línea 210.

Esta forma de colocar espacios en blanco o bien ceros puede ser muy útil en los resultados que se verán posteriormente, al obtenerlas de series o de operaciones con un elevado número de cifras.

